Programming Ideas:

For Teaching
High School Computer Programming

compiled by
Chris Robart

2nd Edition
December 2001

Introduction

This is a collection of ***ideas for*** exercises, test questions, projects, etc. It is ***not*** a collection ***of*** exercises, test questions, projects, etc. This collection is meant to be used when the teacher first begins to design an assignment or test and needs some ideas. This collection is no good to the teacher who is in a panic and needs a string exercise for a class starting in five minutes!

It is assumed that the reader is a computer science teacher who is already familiar with most of the ideas.

The ideas are kept short and somewhat cryptic because the intent is that as many ideas be presented in as small a space as possible. Also they are kept as general as possible so they can be used in a variety of ways (assignments, major projects, quick test questions, etc.) and with a variety of languages (Pascal, Turing, Basic, C, Java, etc.)

The ideas span the range of difficulty from simple ideas that could be used in grade 9 (or earlier) to problems that are more appropriate for university. The problems are primarily classic computer science problems or mathematical, with a liberal sprinkling of fractal geometry thrown in for good measure. Some business ideas are included and of course as many games as possible. The rule I use for inclusion is, "Can I see myself ever using this idea?". There are close to 600 ideas, many with multiple sub ideas.

The major headings correspond to the topics that I have taught in grades 10, 11 and 12. There are ***no sections on graphics, GUI, applets or files***, as these topics can be part of any problem. Most programs could be written with or without a GUI and could use file or keyboard and screen I/O .

I am always searching of new ideas to add to this collection, so any ideas not here would be greatly appreciated. (Corrections, clarifications and other suggestions are always welcome too.)

Thank you and have fun programming.

Chris Robart
crobart@mmhs.ca

Milliken Mills High School
7522 Kennedy Road
Markham, Ont
L3R 9S5
(905) 477-0072

Brief Table of Contents

A Sequence

1. Formatted output
   a. print various messages
   b. menus
   c. shapes
   d. large letters
   e. receipts
   f. cheques
   g. labels
   h. calendars
   i. titles

2. Simple formula
   Given all but one unknown, find the remaining value.

Geometric Formulas:

Rectangle: area = hw
perimeter = 2h + 2w
Parallelogram: area= bh
perimeter = 2h + 2w
Triangle: area = ½bh
Hero's formula:
area = sqrt(s(s-a)(s-b)(s-c))
s=½(a+b+c)
perimeter = a + b + c
Trapezoid: area = ½h(a+b)
Regular polygon: area = ¼nb^2 cos(PI/n)/sin(PI/n)
perimeter = nb
Circle: area = PIr^2
circumference = 2PIr
arc length = rt (in radians)
Box: Volume = lwh
Surface area = 2(lw + lh + hw)
Sphere: Volume = 4/3PIr^3
Surface Area = 4PIr^2
Cylinder: Volume = PIr^{2}h
Surface Area = 2PIrh
Cone: Volume = **a** PIr^{2}h
Surface Area = PIrl
l=sqrt(r^2+h^2)

Mathematics:

Triangle numbers: nth number = n(n+1)/2
$x^n = e^{(n \ln x)}$
law of sines: a/sinA = b/sinB = c/sinC
law of cosines: c = sqrt(a^2 + b^2 - 2abcosC)
sum of arithmetic series:
$t_n = a + (n - 1)d$ (nth term)
$sum = n(a+t_n)/2$
sum of geometric series:

$(ar^n - a)/(r - 1)$ if $r > 1$
$(a - ar^n)/(1 - r)$ if $r < 1$
sum of infinite geometric series:
$a/(1 - r)$ $|r| < 1$
slope of a line m = (y2 -y1) / (x2 - x1)

Physics Formulas:
velocity = d/t
acceleration =v/t (or d/t^2)
force = mass x acceleration
work = force x distance
kinetic energy = $\frac{1}{2}mv^2$
potential energy = mgh
power = work / time
momentum = mv
pressure = force / area
$E = mc^2$
density = mass / volume

Distance to horizon (in km) D = sqrt (0.126*H)
(h is height above ground in centimetres)

Wind Chill Factor (in degrees F):
91.4-(0.288sqrt(v)+0.45-0.19v)(91.4-t)
v is in mph and t in degrees F

Humidex (in degrees F):
0.40(DBT + WBT) + 15
(dry/wet bulb temperatures)

Automobile Tire Pressure:
PV = 0.37m(T + 460)
P = pressure in psi.
V = volume in cubic feet
m = mass of air in pounds
T = temperature in Fahrenheit

Electrical Formulas:
R = Total resistance
Parallel circuit:
$1/R = 1/R_1 + 1/R_2 + 1/R_3$
Series circuit:
$R = R_1 + R_2 + R_3$
V = IR (Ohm's Law)
$P = I^2R = VI = V^2/R$

Business Formulas:
Simple Interest = P x I x T
Compound interest = $A (1 + r/n)^{ny}$
(n = # compounding periods per year)
Present value=income / (1+discount rate) $^{\text{years in future}}$

3. Conversions

Temperature:
Celsius = 5/9 (F - 32)
Kelvin = C + 273.16

Metric:
1m = 39.37in
1mi = 1.609km
1kg = 2.2046lb
1km/hr = 0.6214mi/hr
1kilowatt = 1.341hp

Mathematical
180 degrees = PI radians
degrees (decimal) into degrees with minutes and seconds

General Applications

4. Time calculations
   a. calculate the number of seconds, minutes, hours, days, weeks, years you've been alive.
   b. convert time in seconds to days, hours, min and seconds or reverse

5. Sports calculations
   a. batting average = number of hits / times at bat

6. Marks conversions
   Given a mark on a test and the test total, find the percentage mark.

7. Multiple choice test score
   Calculate the final score on a multiple choice test.
   final = number correct - 1/4(number wrong)

8. Average of 2 or 3 numbers
   Find the average of 2 or 3 given numbers

9. Pet count
   Calculate the total number of pets the user has after asking for the number of cats, dogs, fish, birds, etc.

10. Pulley formulas
    a. calculate the speed of one pulley if there are 2 pulleys connected with a belt:
       RPM2 = diameter1/diameter2 * RPM1
    b. calculate the amount of weight that can be lifted with a multiple pulley system:
       weight lifted = force exerted * number of up ropes

11. Musical scales
    C, C#, D, D#, E, F, F#, G, G#, A, A#, B, C'
    C is 256hz and C' is 512hz, compute the intermediate frequencies if:
    $f_{n+1} = f_n(2^{1/12})$ (even tempered scale)
    also calculate $f_D = 9/8 f_C$ and $f_E = (10/9) f_D$ (natural scale)

12. Making music in Turing

Translate music into Turing: various notes, lengths and octaves can be played. Also make the keyboard into a piano.
(Stephenson 1994, Gr10, Ex)

13. Date for Easter
Given the year calculate the day and month of Easter:

a = year mod 19
b = year div 100
c = year mod 100
d = b div 4
e = b mod 4
f = (8b + 13) div 25
g = (19a + b - d - f + 15) mod 30
h = (a + 11g) div 319
i = c div 4
j = c mod 4
m = (2e + 2i - g - h - j + 32) mod 7
month = (g - h + m + 90) div 25
day = (g - h + m + month + 19) mod 32
(month will be 3 or 4: March or April)
(Carter 1989, p70 and alternate is given in Dyck 1984, p94)

Business Applications

14. Business calculations
a. given the price calculate the taxes and total cost of an item.
b. given the weight and cost per kg, find the total cost.
c. given the sales, cost and expenses, find the profit
(profit = sales - cost - expenses)
d. find the payment on a loan
payment = (rate*principal/N)/ (1 - (rate/(N + 1)) $^{-N*term}$))
(N = annual number of payments)

15. House buying
a. the price you can realistically afford is 2.5 times your gross income.
b. you can afford monthly payments of between 25 and 30% of gross monthly income.

16. Future population estimate
FP = CP (1 + (BR - DR))^{years}
CP is the current population and BR and DR are the birth and death rates.

17. Chocolate chip cookie problem
There are 12 cookies per box (sold at $1.14) and 24 boxes per carton. Left over boxes are sold for 57¢. Remaining cookies are given away free. Given the number of cookies produced, determine the number of boxes, cartons, left over boxes and the total money made.
(Solow 1988, p48)

B Loops

These include problems that require JUST loops. No decisions (other than the exit condition) need to be made.

See also: **For Loops**

1. Arithmetic / geometric sequences
   Generate them or add them up given a, n, d/r. Stop after # terms or after a term exceeds some value.

2. ASCII values
   Print the ASCII numeric value of any key stroke (use a loop and exit when a certain key is pressed)

3. Total / average
   Find the total or average of a set of numbers stop after so many numbers, or after special input value. Example: student marks

4. Password checker
   Exit after the correct password is entered or after x tries.

5. Conversion tables
   Generate tables of equivalent temperatures, weights, speeds, etc. using the conversion factors in the sequence section.

Math Applications

6. Tables of powers, etc.
   Generate tables of powers of x, or a table of the square root, square, and cubes of a range of numbers.

7. Mathematical calculations
   a. GCD
   b. Fibonacci sequence
   c. Factorials
   d. Triangular Numbers (1, 3, 6, 10 ,15, 21, 28, 36, ...)

8. Digits of a Number
   a. find sum of digits
   b. determine the number of digits in a number
   c. find the **digital root** of a number (repeatedly find the sum of the digits until you get a number 0 to 9. Eg: 285: 2+8+5=15: 1+5=6)
   d. print the digits in reverse order
   e. find the product of the digits of a number
   f. find the **digital product** of a number (product of all non-zero digits of the number.)
   g. find the **persistence** of a number. It is the number if times you can multiple the digits (iteratively) before you get a single digit number. Eg the persistence of 467 is 4 because 4*6*7=168 1*6*8 = 48 4*8 = 32 3*2 = 6. 4 steps. The persistence of 5 is 0.
   h. given a number ending with a 6, called A. Let B be A with the 6 moved to the front. Eg A = 4576, then B = 6457. Find all the numbers A (or the first number A), such that B = 4*A.

9. Means:
   a. Arithmetic mean (normal “average”)

= sum of the numbers / n

b. Geometric mean
= product of the numbers / n

c. Harmonic mean
= 1/ [(1/n)(SUM($1/x_i$))]

10. Long division
Calculate x/y with only integers to a given number of decimal places or when it terminates:

```
q = x div y
print q, "."
r = x mod y
while r not= 0
    d = r * 10
    q = d div y
    print q
    r = d mod y
```

11. Unit fractions or Natural fractions
Convert a normal fraction a/b into the sum of unit fractions:
ie: $a/b = 1/n_1 + 1/n_2 + ... + 1/n_k$
Method:

```
while a not= 0
    n = smallest integer >= b/a
    print n
    a = a*n - b
    b = b*n
```

12. Decimal to binary
Convert small integers (< 256) into binary

13. Radicals
Given x and y representing x*sqrt(y), simplify the expression.

14. Series calculations
For either n terms, until a term < e, or sum > x.

a. $1 + ½ + 1/3 + 1/4 + ...$
b. $1 + 1/2^2 + 1/3^2 + 1/4^2 + ...$
c. $1 + ½ + 1/2^2 + 1/2^3 + ...$
d. $4 / (4k - 3)(4k + 1)$
e. $4/(3^{(k-1)})$
f. $1/(k(4k^2 - 1))$
g. $(-1)^{k-1}/((2k)(2k - 1))$
h. $(-1)^{k+1}/k^k$

15. Converging sequences
Determine if the following sequences converge.
(ie: for either n terms, the difference of terms < e or term < e)

a. $(1 + 1/n)^n$
b. $(1 - 1/n)^{-n}$
c. $(n^3 + 4) / (4n^2 + 7n)$
d. $(1 + 2/n)^n$
e. $sqrt(n + 1) - sqrt(n)$

f. $(2n^2 - 4n) / (5n^2 + 10)$

16. Harmonic series
Calculate the smallest number of terms necessary that
$1 + ½ + 1/3 + 1/4 + ... >= x$

17. Golden mean (iteration)
Start r = 1 and iterate:
$r_{new} = 1/(1+r_{old})$
Correct answer is (sqrt(5) - 1)/2

General Applications

18. Volume/Pressure/Temperature relationships
Print a table of values for V2 as T2 varies from -100 to 100 and P2 varies from 100 to 1000 and V1=500, P1=53 and T1=500.
V1*P1/T1 = V2*P2/T2

19. Bouncing ball
A ball is dropped from a height of x. It rises 2/3 of previous height on each bounce. How many bounces until height is less than y?

20. Random walk
A drunk is in the middle of a 10 meter bridge. He is equally likely to take a step forward as backward. Each step is 1 meter long. How many steps, on average, is he likely to take to get off the bridge?

21. Land management
Careless land management causes 1% of the topsoil to erode each year. 9cm of topsoil is too shallow to grow crops on a large scale. If Canada has 30cm of topsoil now, how long will it be before the depth is eroded to 9cm?

22. Animals and food supply
If there are 10 animals in a lab and enough food for 1000 animals, and every hour the number of animals doubles and enough food for another 4000 animals is added, determine when the population exceeds its food supply.

23. Predator/Prey simulation
Given an initial number of foxes and rabbits (F and R), birth and death rates of them (BR, BF, DR, and DF) and the rate at which foxes eat rabbits (E), plot (R,F) on the screen and watch the patterns. The equations are:
R = (1 + BR - DR*R)*R - E*P
F = (1 - DF)*F + BF*P
P = R*F
P starts at 0.
(Algorithm 1.4, p4)

Business Applications

24. Accumulated savings
Print the amount of money in a bank account at the end of each year if the interest rate is r and a fixed amount of money is deposited each year.

25. Simple mortgage program
If the mortgage is for $120,000 at 9.5% per year, and if $24,000 is paid each year, how long before the mortgage is paid off?

26. Simple break even analysis
For a specific cost and revenue equations (eg: C = 1500 + 1.78U and R = 2.03U where U is the number of units produced) use a loop (varying U) to find out where R = C or where $x profit is made. Graphics could be used to plot equation.

27. Maximum profit
If a cinema sells tickets for $5 and averages 100 tickets sold per show, the cost per person is $2. For each reduction in price of $0.25, the number of tickets sold increases by 30 and the cost per person increases by $0.10. Print a table showing the price (ranging from $5 to $3), the number of tickets sold, cost per person, gross revenue, total cost, net profit. Indicate the maximum profit.

C Decisions

These problems may or may not involve loops. (Generally I find that its easier to do loops FIRST than decisions second. Because it gives more problems to choose from!)

See also: **Case / Switch**.

1. Number classification
   even/odd
   positive/zero/negative

2. Find the absolute value of a number

3. Determine if a is a multiple of b

4. Sort 2 or 3 items
   Given the items in random order, print in sorted order: allow for duplicates

5. Counting between 2 numbers
   Given two number, count starting from the smallest to the largest number.

6. Count passes/failures in a set of marks

7. Above or below determination
   Given a series of values, determine the number of times the values go above/below a given value

8. Range determination
   given a series of numbers count the number of numbers within certain ranges (eg 0-49, 50-59, 60-70, etc.)

9. Find highest/lowest number from a given set

Math Applications

10. Palindrome
    Determine if a given number is a palindrome.

11. Palintuples
    Find all numbers which are multiples of their reversals.

12. Divisibility tests
    a. a number is divisible by 3 (or 9) if and only if the sum of its digits is 3 (or 9).
    b. a number is divisible by 11 if and only if the alternating sum:
       $a_k - a_{k-1} + a_{k-2} - .. \pm a_0$ is divisible by 11.
       (a_k is the most significant digit and a_0 the units digit.)
    c. a number is divisible by 7 if and only if it passes the following test:
       take last digit and double it
       take the rest of the digits and subtract the double last digit
       repeat until result is 7, -7 or 0.
       Eg: 10976 -> 1097-12 = 1085 -> 108-16 = 98 -> 9-16 = -7
       49 -> 4 - 18 = 14 -> 1 - 8 = -7

21 -> 2 - 2 = 0

13. Calculate pi using random numbers (Monte Carlo Algorithm)
Generate a series of n random numbers, x and y, between 0 and 1. For each pair (x,y), calculate the distance from (0,0). If distance < 1, add 1 to a counter. Then pi . (counter/n)*4

14. Hailstone numbers
For a given n, iterate using the following rule:
if n is even n = n/2, else n = 3n + 1
Stop after the sequence 4, 2, 1 occurs. Will all n lead to this sequence?
Calculate the length of the sequence for a given n.
(Algorithm 1.1, p5, Giblin 1993, p32)

15. Quadratic equation
Determine if a quadratic equation has real roots (distinct or equal) or no real roots. Print the roots if they are real.

16. Multiply 2 numbers as if by hand
Given 2 integers, multiply them showing all partial products.
eg:
```
1025   (output format is critical)
  12
 ----
 2050
1025
-----
12300
```

17. Primes and factors
a. determine if a number is prime
   i. division method
   ii. witness method (Dewdney 1989, p310)
b. determine the prime factors of a number

18. Crossing lines
Given the equations of two lines, ($y = m_1x + b_1$ and $y = m_2x + b_2$) determine if the two lines:
are parallel: $m_1 = m_2$ and b_1 not= b_2
are the same: $m_1 = m_2$ and $b_1 = b_2$
intersect, and if so compute that point and state the quadrant (or origin) that it is in.

General Applications

19. Leap year
Determine if a year is a leap year (include the check for century years: they are leap years only if divisible by 400.)

20. Elementary arithmetic drill
Time how long it takes for the user to answer x addition (subtraction, multiplication or division or a combination of these) questions. (Use random numbers to generate the questions.) The user can either input the difficulty level (levels reflect the size of numbers used: 0-5, 0-9, 0-20, 0-100, -10...+10, etc.) or start the drill at the easiest level and increase one level after x correct answers in a row.

21. Age determination

Determine the age in years of a person, given the person's birth date (day, month, year) and knowing the current date.

22. A balanced lever
A lever is balanced if the force * distance from the fulcrum is the same on both sides of the fulcrum. Given the two forces and distances, determine if the lever is balanced.

23. Weather problem
Input a series of daily weather values: rainfall, min/max temperature and pollution count then determine the following:
    first rainy day with max temperature > x
    # days with max - min > x
    x warmest days
    x consecutive days with most rainfall
    etc.
(Hume 1990, p69)

24. Truck variation of the cashier's problem
Assume the capacity of a truck is 1837 kg. How many items (of each) weighing 625, 125, 25, 5, and 1 kg can be loaded on the truck?

25. SIN check
    a. double 2, 4, 6, 8 digits and add the digits
    b. add the 1, 3, 5, 7 digits
    c. add the results of a and b.
    d. take 10 - units digits of the result in step c, and this should equal the 9th digit.
(Carter 1989, p213)

Business Applications

26. Cashier's problem
This is a decision problem and not simply a sequence problem if you insist that zero dimes not be printed, nor 1 dime**s** (ie: make it grammatically correct).

27. Parking garage
Given a length of stay (in minutes) determine the parking charge: the rate structure can vary, but there should be a minimum charge, maximum charge and use the idea of "part thereof" (eg: typically 31 min will cost the same as 59 or 60 min). May also include such things as early bird specials, and holiday flat rates.

28. Simple banking machine
Enter a balance then a series of transactions (type: deposit, withdrawal, bill payment and amount) and update the balance and print a receipt. Continue until the customer enters a "stop" transaction.

Games

29. Best poker hand
Given 5 cards (values and suits), sorted, determine what is the best poker hand: royal flush, straight flush, 4 of a kind, full house, flush, straight, 3 of a kind, 2 pair, pair, high card.

30. Guessing game (high / low)

The computer picks a number between 0 and x, then the user tries to guess the number in the fewest guesses. After each guess the computer will tell the user "high" or "low". (Ask the students what is the optimum strategy and maximum # guesses it should take for various x.)

31. Russian roulette
Include some fun comments to the user. 1/6 chance of loosing!

32. Simple heads/tails guessing game
Ask the user to guess what the result of the next flip of a coin is. Keep score. Another version is to play even or odd (ie. 2 heads/2 tails or 1 head and 1 tail), the user could flip a coin and so could the computer and either the user or computer could guess the outcome.

33. Paper/Rock/Scissors game
Computer and user choose one of the three and compare:
paper covers rock
rock breaks scissors
scissors cuts paper

34. Poison penny (a very simple version of Nim)
x pennies are laid out (x > 3). Two players alternate taking 1 or 2 pennies. Whoever takes the last penny loses. (There can be two players or one player versus the computer.)
(Drake 1988, p112)

35. Simple craps game
A player rolls 2 dice. If 7 or 11 are rolled he wins. If 2, 3 or 12 are rolled he loses. Otherwise the number is his "point". If he can it again before a 7 he wins.
Simulate 1000 games of craps and calculate the losing %. Then continue to simulate games and track the losing %. Stop when the losing % doesn't differ by more than 0.0005 from one time to the next.

36. Mars rover
Given input in the form:

1 x	(Move x units forward)
2	(Turn right 90)
3	(Turn left 90)

Calculate the distance needed to return home and give the shortest sequence of commands to get home (back where the rover started.)
(CCC 1998)

37. The Tortoise and The Hare
Have the tortoise and hare complete a race of x units long. Use a random number to determine the distance each travels per unit time.

Tortoise:	50%	3 squares ahead
	20%	6 squares back
	30%	1 square ahead
Hare:	20%	no move
	20%	9 squares ahead
	10%	12 squares back
	30%	1 square ahead
	20%	2 squares back

Include fun commendatory on the race.
(Deitel 1999, p316)

D Case/Switch

These are decision problems that involve an extended (more than 3) “else if” structure.

See also: **Decisions**

1. Marks / letter grades
   Convert marks to letter grades or comments such as “excellent”, “very good”, etc.

2. Calculate grade point averages
   Given the grade (A, B, C, D, or F and the number of credits for each course, calculate the grade-point average: (# credits x mark) / total credits. (A = 4.0, B=3.0, C=20, D=1.0)

3. Count marks in given ranges

4. Maximum days in a month
   determine the max number of days in a month. (Jan - 31, Feb - 28, March - 31, etc.)

5. Vote counting
   Given a series of votes for different parties, maintain totals for each party. (Display continuously / graphically)

6. General menu
   Devise any sort of menu and display an appropriate message for each choice.

7. Simple calculator
   Input number, operator and number (5+2) and perform the operation (operations can be +, -, *, /, ^)

8. Student classification
   Classify students based on number of credits:

freshman	<5
sophomore	5-9
junior	10-14
senior	>14

9. Convert digits to words (eg 9 to “nine”)

10. Origin of SIN numbers
    Determine the region of Canada where the SIN was issued:
    1-Atlantic, 2-Quebec, 4-Ontario, 6-Prairies, 7-BC.

11. Ticket classes
    Given the age of the customer, determine the class: child, student, adult or senior. The age breaks can be set to whatever you like.

12. Acid rain
    Given the pH of the rain water, print an appropriate message:

6.5 - 7.5	neutral
< 6.5	too acid
> 7.5	too alkaline

13. Astrological signs
Determine the astrological sign given the month and day of the person's birthday:

Aries	Mar21-Apr19	Libra	Sep23-Oct23
Taurus	Apr20-May20	Scorpio	Oct24-Nov21
Gemini	May21-Jun21	Sagittarius	Nov22-Dec21
Cancer	Jun22-Jul22	Capricorn	Dec22-Jan19
Leo	Jul23-Aug22	Aquarius	Jan20-Feb18
Virgo	Aug23-Sep22	Pisces	Feb19-Mar20

14. Dewey decimal library classification
Given the number of the book, print the classification:

000-999	General	500-599	Pure Science
100-199	Philosophy	600-699	Technology
200-299	Religion	700-799	Arts
300-399	Social Sci.	800-899	Literature
400-499	Language	900-999	History

15. Temperature classification
Given a temperature, print an appropriate message:

< -18	very cold
-18 to <0	cold
0	freezing point of water
1 - 10	very cool
11 - 20	cool
21 - 30	warm
31 - 40	hot
41 - 99	very hot
100	boiling point of water

16. Birthstones
Given a month, print the appropriate birthstone
(Drake 1988, p244)

17. Character classification
Determine if a character is alpha, numeric, punctuation, arrow or F key.

18. Dvorak Simplified Keyboard (DSK)
Convert each letter of the standard QWERTY keyboard into DSK.
(Drake 1988, p205)

19. Compass Bearings
Convert a bearing (0 - 359) to the nearest N, E, S, or W. (If a bearing is half way between the points, pick N or S as appropriate.)

20. Very simple assembly language
Perform the operations of a very simple assembly language. Assume only one register. The commands are: load, add, subtract, print and stop. Eg:

load 5 add 2 print stop

would produce 7 on the screen.

Math Applications

21. Triangle classification
Input the lengths of the 3 sides and determine if: equilateral, isosceles or scalene and also if it is right, obtuse or acute (use Pythagoras) or if it is a triangle at all.

22. Quadrilateral classification
Input the lengths of the sides and angles, in order and determine if it is a square, rhombus, parallelogram, rectangle, kite, trapezoid, or general quadrilateral.

Business Applications

23. Gas meter
Reading a gas meter and calculating the bill (the meter is a 4 digit meter and two readings are taken: note if the first reading is 9980 and the second 0020, then 40 was used)

first 10	$6.59 (minimum bill)
next 20	23.73¢ per unit
next 55	22.71¢ per unit
next 85	21.78¢ per unit
over 170	20.85¢ per unit

24. Postage problem
Given the weight of a letter in grams, determine the cost of the postage:

up to 30g:	48¢
up to 50g:	70¢
up to 100g:	90¢

18¢ for each 50g or less after that.

25. Credit card application evaluation
Based on the input values, points are obtained. Based on the total points, various actions are taken.

		Points
Age:	<20	-10
	21-30	0
	31-50	20
	>50	25
At current address:	<1yr	-5
	1-3yr	5
	4-8yr	12
	>8	20
Annual income:	<$15000	0
	<$25000	12
	<$40000	24
	>$40000	30
At same job:	<2yr	-4
	2-4yr	8
	>4yr	15
Actions:	<20 points	no card
	21-35	card with $500 limit
	36-60	card with $2000 limit
	>60	card with $5000 limit

26. Weekly pay slips
Generate weekly pay slips. As an example: input could be hourly rate and number of hours worked, then calculate the gross pay (double time after 40 hours) and deductions based in tax category

A	0%
B	10%
C	20%
D	29%
E	35%

and United Way deduction of $10 (input a Y/N code for this).

Games

27. Cards
Given a number 1 to 52, convert this into a card (ie suit and value) as in "Queen of hearts" or "5 of diamonds".

28. Sports divisions
Convert sports divisional codes to full words.
example: MB midget boys
MG midget girls
JB
JG
SB
SG senior girls

29. Football
Given two team names, continue to ask for a name and type of score and keep track of total points, at the end declare a winner

touchdown	6points
field goal	3
convert	1

30. Golf scores
Given the par and score on each hole (9 or 18), state if it was a hole-in-one, eagle, birdie, par, boogie, or double boogie and give the final score (in terms of par). (Note some scores have no special name)

31. Rugby
Same idea as for football

32. Basketball
Same idea as for football (foul shot - 1pt, normal - 2pt, long - 3pts)

33. Snooker
Edit colors: red alternates with a color, then colors in order of points. red - 1, yellow -2, green - 3, brown - 4, blue - 5, pink - 7, black - 8.

E For Loops

See also : Many **function / procedure** problems are for loop problems.

1. Arithmetic/Geometric sequence
   Print and/or find the sum of a arithmetic/geometric sequence.

2. Student / Class averages
   For each of x students, get y marks and calculate each student's average and the class average (double loop problem)

3. ASCII values
   Print the character for each ASCII value 0-127 (if possible).

4. 2D tables
   a. times/addition tables
   b. table of square roots, squares, cubes, etc.
   c. any function of 3 variables

5. Digit pyramid
   Generate the following pyramid of digits:

```
          1
         232
        34543
       4567654
      567898765
     67890109876
    7890123210987
           ...
```

6. Star shapes (character graphics)
   Given the dimensions of a rectangle or square, draw the corresponding shape with stars.

7. Simple graphics problems
   a. simple shapes (made with stars, letters, etc.):
      diagonals, diamonds, squares, triangles(4 orientations), pyramids, letters, parallelograms, trapezoids, etc. (filled or hollow) given the size.
   b. histograms
      given some sports data (ie: player and RBIs) plot it as a histogram
   c. animation:
      i. bouncing balls (Algorithm 2.3, p5)
      ii. simple shooting games: given the angle and force shoot something at a target, if the formula for motion is a straight line you can make a golf game, if the formula is a parabola you can make cannon ball type games. (Dewdney 1993b, p71)
      iii. launching satellites into orbit from the space shuttle (Algorithm 2.4, p5)
      iv. circular rings moving out or overlapping, etc.

Math Applications

8. Mathematical formula
   a. Fibonacci sequence
   b. factorials

c. triangular number

9. Powers
   a. list powers of a given number
   b. calculate x^n given x and n

10. Mathematical functions
   a. Calculate a table of values for any mathematical function
   b. Graph the function either in pixel or character graphics, given a range.

11. Prime and composites
   a. find all twin primes, up to x.
   b. find all pairs of primes such that the larger is 1 more than twice the smaller (eg: 3 and 7), up to x.
   c. for a given k >= 2, find the smallest k consecutive composite #'s. (There must be such a sequence since the k-1 numbers k!+2, k!+3,..., k!+k are all composite.)

12. Number theory calculations:
   a. triangle numbers (1, 1+2, 1+2+3, ...)
   b. find perfect numbers. A perfect number= sum of its proper divisors (including 1 and excluding itself).
   c. determine if a number is an abundant/deficient number. An abundant number is less than the sum of its proper divisors. (A deficient number is the reverse.)
   d. find all 3 digit numbers that are equal to the sum of the cubes of their digits. (eg: $1^3 + 5^3 + 3^3 = 153$)
   e. find all four digit numbers for which the square of the sum of the first two digits and the last two digits equal the number itself. (eg: $3025 = (30 + 25)^2$)

13. Consecutive series
Find the series of consecutive positive numbers whose sum is 10,000
Find all such series of consecutive positive numbers that sum to 10,000 (or any other given number).
(Wiesenburg1987,p112)

14. Mathematical series calculations

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 + \ldots$$

$$\text{sqrt}(1+x) = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{x^3}{16} - \ldots$$

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

$$\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots$$

$$\ln(1+x) = \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \ldots$$

$$PI = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots\right)$$

$$PI = \sqrt{6\left(1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \ldots\right)}$$

$$PI = 4 * \frac{2}{3} * \frac{4}{3} * \frac{4}{5} * \frac{6}{5} * \frac{6}{7} * \frac{8}{7} * \ldots$$

15. Triangular and square numbers
Generate the sequence of fractions:
$\frac{1}{0}\ \frac{1}{1}\ \frac{3}{2}\ \frac{7}{5}\ \frac{17}{12}\ \frac{41}{29}, \ldots$
To generate, each denominator is the sum of the previous numerator and denominator and each numerator is the current denominator plus the previous denominator.
From this sequence can be generated all the numbers which are both triangular and square. The values: (numerator * denominator)^2 (Eg. 1, 36, 144, are the first 3 numbers which are both square and triangular.)
Also the triangular root of each can be found by finding the lesser of numerator^2 and 2*denominator^2 . (eg in the case of 36, it is lesser of 9 or 8, which is 8. That is: the 8th triangular number is 36)

General Applications

16. Radioactive isotope table
Calculate the amount remaining of a radioactive isotope after each year: amount remaining = initial amount * $C^{(year/H),}$ where $C = e^{-0.693}$ and H is the half-life of the isotope

17. Cutting paper
If a piece of paper has an area =1m^2 and a thickness = 0.090mm, and is cut in half and the pieces stacked together, and then cut again, etc., what will the area and thickness be after 40 cuts? (Or when will the stack reach a height of x?)

18. M random numbers from a list of N numbers
Given 2 numbers M and N with M < N, print a sorted list of M numbers randomly selected from the range 1..N.

```
selected = m
remaining = n
for i = 1 to n
        if rand(0,1) < select/remaining then
                print i
                selected = selected - 1
        remaining = remaining - 1
```

(Bentley 1986, p117)

19. ZigZags
Start with a horizontal line of 100 units and 2 angles a and b and a length L (>0). At the end of the horizontal line, draw another horizontal line of length L. Now start the zigzag process by drawing lines of length 100, L, 100, L, ... with anticlockwise angles of a, b, 2a, 2b ,3a, 3b, ... These zigzags will close up. (Use 2 different colors)
(Giblin 1993,p33)

Business Applications

20. Compound interest (without formula)
    a. the world's population was 5 billion in 1987. At a growth rate of 1.6% per year, what will the population be in 2087?

b. how much is $24 worth today if it was invested at x% in 1627? (The Manhattan Problem)

21. Investments
Make a table for the amount of money one needs to invest now, to get $1000 in the future. Each row can be a different number of years (n) and each column could be a different interest rate. The formula is:

$$A = \$1000 / (1 + r)^n$$

22. Mortgage problem
Create a table showing the monthly payments needed for a mortgage of x amount. The rows can be different interest rates and the columns can be different amortization periods (years). The method is:

n = years * 12
$i = (1 + (\text{rate}/200))^{(1/6)} - 1$
$f = 1 / ((1 - (1 + i)^{-n})/i)$
r = cost * f

cost is the mortgage amount, i is the effective interest rate per month, and r is the monthly payment.

With this information, you could also determine the amount of interest and principal paid off each month (eg. the amount of interest = outstanding principal * i, the principal paid is = r - interest paid, and the new outstanding principal = outstanding principal - principal paid.)

Fractal Applications

23. Chaos game (Sierpinski's Triangle)
a. given the coordinates of 3 corner points and a randomly selected point, choose a random number 1 to 3. Depending on that number plot a point half way between the current point and the corresponding corner. Then choose another random number and from the current point go half way to corresponding corner. Repeat many times.
b. extend this idea to include any number of corner points, (allow the user to select them, randomly choose them or put them in a circle at the far edges of the screen) and allow the user to choose if on each time you go half way or some other factor.

24. Coin toss game and fractals
Toss a coin many times and for each head add one to a counter and for each tail subtract one from the counter. Display the counter versus the number of toss on a graph. Track the distances between zeros of the graph and plot the results on a log-log graph
(Lewis 1990, chap 5)

25. Lorenz attractor
Iterate the following equations and plot the x and y coordinates.

$xnew = x + 10h(y - x)$
$ynew = y + h(28x - y - xz)$
$znew = z + h(xy - 8/3z)$

Let h= 0.01 and start x,y and z = 0.06 (then try x and y = 0.06 and z = 0.06001). Repeat for 400 points or until the user hits a key.
(Algorithm 1.2, p8)

26. Henon's attractor
Iterate and plot the points of the following:

$x_{n+1} = y_n - ax_n^2 + 1$
$y_{n+1} = bx_n$

Let a = 7/5, b=3/10 and $x_0=y_0=0$.
(Hofstadter 1985, p384, Peitgen 1991b, p274)

27. Verhulst (bifurcation) diagrams and Lyapunov exponents
    a. iterate the function rx(1-x) for various r. After a while plot the points. This will give the classic Verhulst diagram.
    b. this can be taken this one step further and calculate the Lyapunov exponent for each r, and alternate r between two numbers (a and b), then plot the point (a,b) using the exponent to determine color.
    (Dewdney 1993b, p100, Devaney 1990, p26, Devaney 1990, p70)
    c. another option is to only plot every odd (or even) iteration
    (Algorithm 2.3, p4)

28. May's equation (and others): bifurcation diagrams
    a. iterate $rx(1+x)^b$, let b=5, vary r from 0 to 125 in steps of .31 and set the initial x = 0.9.
    (Algorithm 1.2, p9)
    b. use $x = 4\sin(r)x(1-x)$, r from 0 to 4 (Algorithm 1.4, p2)
    c. use $x = x^2 - a^2x - ax + a$

29. Gingerbread man
    a. iterate the equations
    $$x_{new} = 1 - y + |x|$$
    $$y_{new} = x$$
    Start (x,y) at some point such as (-0.1, 0)
    (Algorithm 3.1, p15)
    b. try this variation
    $$x_{new} = 1 - y + k|x|$$
    $$y_{new} = x$$
    Vary k from -2 to 2 and use different start points.
    (Algorithm 2.2, p3)

30. Mandelbrot set creation
    a. generate the Mandelbrot set. The equation is $x = x^2 + c$, where all variables are complex numbers. Let each pixel represent a different value of c. Start x at 0 and iterate the function until either the $|x|>2$ or after 1000 (or other large number) times. Color the pixel on the basis of count. (If count reached the maximum, then its inside the Mandelbrot set and color it black, otherwise for different ranges of count use different colors or try color = count mod 16) The full Mandelbrot set is from -2 to 0.5 (horizontal direction) and 1.25 to 1.25 (vertical direction).
    b. Boundary Scanning Method (BSM). Gives a good outline of the Mandelbrot set.
    (Devaney 1990, p123)
    c. Distance Estimator Method (DEM). Gives the best pictures of the Mandelbrot Set, but is more complex than other methods.
    (Peitgen 1988, p192)
    d. instead of using the equation $x = x^2 + c$, try the family of equations $x = x^n + c$ where x and c are complex, and n is real.
    (Algorithm 1.7, p4)

31. Julia set creation
    a. same procedure as for Mandelbrot Set creation, except c is fixed, and each pixel represents a different starting value of x. (Note: there is only one Mandelbrot set, there is a infinite number of Julia sets; one for each value of c).

b. Inverse Iteration Method (IIM). Based on plotting points in the orbit of the inverse function x = sqrt (x - c)
(Devaney 1990, p102, Peitgen 1991b, p413, Peitgen 1988, p152)

c. Boundary Scanning Method (BSM). Gives a good outline of the Julia Set.
(Devaney 1990, p111)

d. Distance Estimator Method (DEM). Best method to generate the Julia Sets, but is more complex than other methods.
(Peitgen 1988, p192)

32. Wallpaper
Nested for loops and simple formula iteration produces some interesting graphics (like certain types of wallpaper).
(Dewdney 1988, p15)

F Functions

See also: All previous sections (many problems can be "functionized").
Procedures if doing C or Java

1. Conversions
   Temperature, time, metric, etc.

2. Largest/smallest of 2 or 3
   Return the largest (or smallest) of 2 or 3 given numbers

3. Statistical calculations
   a. min / max (range)
   b. mean, median, mode
   c. standard deviation

4. Logical functions
   Write logical functions for all of the logic gates (NAND, NOR, XOR and XNOR)

5. Conversion of lower case to / from upper case

6. General menu
   Return a choice from a menu

Math Applications

7. Mathematical calculations
   a. GCD, LCM, and Relatively prime
   b. factorials
      permutations: P(n,r) = n!/(n-r)!
      combinations: C(n,r) = n!/[r!(n-r)!]
   c. absolute value
   d. powers (x^n)
   e. calculating e^x, sin(x), etc.

8. Number theory problems
   a. determine if a given number is a **triangle** (or **perfect**, etc.) number
   b. **friendly** numbers: two numbers are friendly if each one is the sum of the divisors of the other (include 1, but not the number itself in the divisors). Eg: 220 and 284 are friendly: 1+2+4+5+10+11+20+22+44+55+110=284 1+2+4+71+142=220

9. Digits of a number
   a. given a number and a position, return the digit of the number in that position (counting from the right), eg: 8247 and 3 would give 2 as a result.
   b. find the sum of the digits of a number
   c. find the number of digits of a number
   d. find the digital root of a number (ie: find the sum of the digits of a number, repeatedly until the sum is between 0 and 9)
   e. find the multiplicative digital root of a number (ie: find the product of the digits of the number, repeatedly until the product is between 0 and 9. eg: 79; 7*9=63; 6*3=18; 1*8=8; therefore the multiplicative digital root of 79 is 8)

f. find the multiplicative persistence of a number. P(x) (The number is steps needed to find the multiplicative digital root: eg: the multiplicative persistence of 79 is 3.)
g. for any n, find smallest x such that P(x) = n.
h. determine if a number is a palindrome
i. determine if a number is a Niven number (a number is a Niven number if it is divisible by the sum of its digits).

10. Happy numbers, Perfect numbers and Practical numbers
a. determine if a number is happy. It is happy if there is a sequence obtained by summing the squares of the digits of the number, repeatedly, converges to 1 in fewer than 15 steps. Eg 23 is happy because $2^2+3^2 = 13$, $1^2+3^2=10$, $1^2+0^2 = 1$ (3 steps)
b. determine if a number is practical. It is practical if it is neither happy nor perfect.
(Brown 1983, p233)

11. Sum of divisors, amicable and perfect numbers
a. find the sum of the divisors of n using the brute force method from the definition: ó(n) = sum of all x such that 1<=x<=n and x divides into n.
b. find ó(n) using the following: for each prime which divides n, calculate the total $1+p+p^2+p^3+...$ as long as the power of p divide into n.
c. define s(n) = ó(n) - n. Then iterate s(n) for a given n (ie: calculate s(n), s(s(n)), etc.) Do all sequences end with 1? or are some periodic?
d. find amicable (friendly) pairs of numbers m and n. m and n are amicable pairs if mnot=n and ó(m) = ó(n) = m+n (or s(n)=m and s(m)=n).
e. find all perfect numbers less than x. A number n is perfect if ó(n)-n = n (or ó(n) = 2n). If ó(n)<2n then n is deficient and if ó(n)>2n it is abundant.
(Giblin 1993, p164)

12. Jack Sprat Numbers
Determine if a number is a Jack Sprat number. A number n is **fat** if the sum of its divisors > 3n. A number n is **lean** if the sum of its divisors = n+1 (ie. prime). A number n is a **Jack Sprat** number if n is lean and n+1 is fat.
(SMSU pp#34)

13. Palindromes by reversal
Given a number, try to form a palindrome from it using the idea: 87+78=165, 165+651=726, 726+627=1353, 1353+3531=4884. Count the number of steps it takes. (stop if the number becomes too large.) Make a table of results showing the number of steps versus starting number.

14. Reversal multiples
Determine if a number is a multiple of its reversal.

15. Square Twins
A twin number is formed by writing a number twice, eg 8181. Find the smallest square twin.

16. Prime numbers
Determine if a given number is prime

17. Integer logarithms
Find the integer logarithm given a number and the base. The integer log of a to the base b is the largest integer x such that $b^x <= a$.

18. Log sums

for a given n calculate $[\log_2 1] + [\log_2 2] + [\log_2 3] + ... + [\log_2 n]$
([x] means largest integer <= x and $\log_2 x = y$ means $2^y = x$

19. Russian multiplication
Given two numbers multiply them using the following method. Successively divide the smaller number by 2 (ignore any remainder) until the quotient is 1 and multiple the larger number by 2. Add to a total only those multiples of the larger which correspond to an odd quotient of the smaller. Eg, 42 * 35

35	42
17	84
8	168
4	336
2	672
1	1344

Then the total is 42+84+1344 = 1428!
(This process is interesting in that all multiplications can be reduced to simple multiplications and divisions of 2.)

20. Linear equations
Find root of linear equation

21. Analytic geometry
a. find the distance between two points:
$d = \text{sqrt}((x_2 - x_1)^2 + (y_2 - y_1)^2)$
b. find the slope of the line given 2 points on the line:
$m = (y_1 - y_2) / (x_1 - x_2)$
c. find the distance from a point to a line (in general form: $Ax + By + C = 0$)
$d = (Ax_1 + By_1 + C) / \pm \text{sqrt}(A^2 + B^2)$

22. Limits of functions
a. input a and h and calculate f(a+h) for ever decreasing values of h (ie: each time though the loop, set h = h /10). Print a table of values showing the results. (Also use a-h and f(a-h).)
b. find the derivative of a function using the above method and [f(x+h) - f(x)]/h

23. Numerical methods
a. finding roots
i. midpoint method
start with two points a and b such that f(a)*f(b) < 0. Then calculate m = (a+b)/2 and if f(a)*f(m) < 0 then set b = m if not set a = m and repeat. Stop when |a-b|<e or f(m)<e.
(Hume 1990, p277)
ii. secant method
(Hume 1990, p277)
iii. Newton's method
start with an initial guess x, and then calculate x = x - f(x)/f'(x). Iterate until f(x)<e or the difference between values of x is very small.
b. finding square roots
Heron's formula
$r_0 = x/2$
$r_{n+1} = \frac{1}{2}(r_n + x/r_n)$
(Hume 1990, p279)
c. finding cube roots

$r_0 = x/3$
$r_{n+1} = 1/3(2r_n + x/r_n^2)$

d. finding areas
For the following: a and b are the limits of integration, a<b and h=(b-a)/n
i. Midpoint
area = hSUMf(x_k) when x_k = a+(k-.5)h, k=1..n
ii. trapezoidal method
area = h/2[$f(x_0)+2f(x_1)+2f(x_2)+...+2f(x_{n-1})+f(x_n)$]
x_0=a and x_n=b and x_k=a+hk
(Hume 1990, p285)
iii. Simpson's rule
area = h/3[$f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+...2f(x_{n-2})+4f(x_{n-1})+f(x_n)$]
x_0=a and x_n=b and x_k=a+hk
(Hume 1990, p286)
iv. Monte Carlo method
(Gottfried 1990, p240)

24. Inverses (division without division!)
To find inverse of c, start with an initial guess x such that: (2 - cx) > 0 and then iterate:
$x_{new} = x_{old}(2 - cx_{old})$
Stop when $|cx_{new} - 1| < e$
(Borse 1985, p90)

25. Power of a prime
a. calculate the power of a prime p which divides n!, given both n and p. The answer = [n/p] + [n/p^2] + n/p^3] + ... until $p^r > n$. [x] = greatest integer <= x.
b. use the function in part a to calculate the power of p which divides the binomial coefficient C(n,r).
c. use the function in part a to calculate the number of zeros at the end of 1000! or C(1000,353).
(Giblin 1993, p26)

26. Power algorithm: find $r = a^n \bmod m$

```
r = 1
while n > 0
        d = n mod 2
        if d = 1
                r = (a*r) mod m
        a = (a*a) mod m
        n = (n-d) div 2
```

(Giblin 1993, p90, Denning 1982, p 39)

27. Euler's function (or totient function)
a. for a given n, calculate Ö(n) = number of integers x such that 1<=x<=n and GCD(x,n)=1.
b. find all n such that Ö(n) = Ö(n+1) or Ö(n) = Ö(n+3) for n < 10000.
c. iterate Euler's function: that is given an n find Ö(n), Ö(Ö(n)), etc. The value should eventually become 2.
(Giblin 1993, p116)

28. Primitive root mod n
a. given numbers a and n determine if a is a primitive root mod n. A number a with GCD(a,n) = 1 and the $ord_n a$ = Ö(n) is called a primitive root mod n.

b. find all primes p < 100 for which the $ord_p 10 = p-1$ (ie. 10 is a primitive root mod p)
(Giblin 1993, p121&131)

29. Divisor function
Find the value of d(n) = number of x such that 1<=x<=n and x divides into n. Use the definition to calculate d(n).
(Giblin 1993, p158)

General Applications

30. Random number generation: Linear-congruential method:
seed = (B * seed + 1) mod M
Seed can be initialized using the system clock. M should be very large and probably a power of 10 or 2. A good choice for B is 1 digit less than M and ending with the digits ...x21 where x is even.
(Sedgewick 1988, p511)

31. Date problems
a. determine if a given date is valid
b. determine if a year is a leap year
c. determine the day of the week for any given date. (Zeller's congruence).
(Collens 1976, p182)
d. determine the number of days between 2 dates (there is a formula!), along with other date functions (valid date, conversions to/from Julian, Zeller's formula, etc.)
(Kochan 1983, p160)

Business Applications

32. Interest calculations
a. calculate simple interest
b. calculate compound interest (using the formula or the simple interest function.)

33. Depreciation calculations
For an item costing x, with an expected lifetime of n years, calculate the depreciation on the item in a given year.
a. straight-line method
each year the item loses x/n of its value.
b. double-declining balance
amount of depreciation in a given year is (2 * current value) / n
c. sum of the year's digits method
amount of depreciation in year j is x times
((n - j) + 1) / (n*(n + 1) / 2)
(Solow 1988, p320)

34. Future value of monthly payments
Given a fixed amount of money deposited each month A, for n years, at an interest rate i,
$F = 12A[((1 + i/m)^{mn} - 1) / i]$
(where m is the number of compounding periods: 1 for annual, 2 for semi-annual, 4 for quarterly, 12 for monthly)
If the compounding periods are shorter than the payment periods (ie. compounded daily) then:
$F = A[(1 + i/m)^{mn} - 1) / (1 + i/m)^{m/12} - 1)]$
If the amount is compounded continuously,

$$F = A\,[(e^{in}-1) / (e^{i/12} - 1)]$$

Make tables with fixed A and n and vary m and i, or keep A and i fixed and vary m and n. (Gottfried 1990, p318)

G Procedures

These are “void” functions!

See also: All previous sections (many problems can be “procedureized”).

1. Hit any key to continue
   Write a simple procedure which will wait until the user hits any key.

2. Swap 2 numbers

3. Sort 2 or 3 numbers
   Rearrange them: don’t print

4. Risk
   The attacker rolls 3 dice and the defender rolls 2 dice. The largest dice of the attacker and defender are compared. If the attacker’s dice is greater then the defender’s dice then the defender loses an army, other wise the attacker loses an army. The same thing is repeated for the second largest dice. Repeat this 1000 times and determine the winner.

5. Shape graphics
   Print x stars in a line. Use this in other procedures to create squares, triangles, letters, etc. given the size.

6. Pixel drawing program
   Create a series of procedures which draw different shapes (rectangles, arcs, triangles, circles, etc. or perhaps, doors, windows, walls, roofs, etc) given location and sizes. Use these to draw complete pictures.

Math Applications

7. Quadratic / cubic equations
   Find the roots

8. Split number
   Given a number (real), split it into its whole and fractional parts

9. Statistical graphics
   Drawing bar charts, pie charts, etc.

10. Analytic Geometry
    a. given two points on the line, find the equation of the line (ie. return m and b in the equation)
       $y = mx + b$
       where $m = (y_1 - y_2) / (x_1 - x_2)$
       $b = y_1 - mx_1$
    b. convert rectangular coordinates to/from polar coordinates:
       $x = r\cos(t)$
       $y = r\sin(t)$
       or $r = \text{sqrt}(x^2 + y^2)$
       $t = \tan^{-1}(y/x)$

11. Mathematical function graphing package
Include plotting the function(s), drawing the axis, setting the scales, etc.

12. Non-function graphs
Plot graphs of non functions:

cardioid	r = a(1+cos(t))	0<=t<=2PI
3-leaved rose	r = asin(3t)	0<=t<=PI
4-leaved rose	r = asin(2t)	0<=t<=PI
limacon	r = b + acost	0<=t<=2PI
spiral	r = at	0<=t<=?
hyperbolic spiralrt = a		0<=t<=?
lemniscate of Bernoulli	$r^2 = a^2\cos 2t$	0<=t<=PI

cissoid of Diocles	$y^2 = x^3/(2a - x)$
astroid	$x^{2/3} + y^{2/3} = a^{2/3}$
witch of Agnesi	$x^2y = 4a^2(2a - y)$
strophoid	$y^2 = x^2(a + x)/(a - x)$

cycloid	x = a(t - sin(t)), y= a(1 - cos(t))
trochoid	x = at - bsin(t), y = a - bcos(t)

(Must convert polar to rectangular coordinates.)

Lissajous figures:
x = Rsin(At)cos(Bt)
y = Rsin(At)sin(Bt)
R, A and B are constants and let t vary from 1 to 1000.
(Sci Am Jan91, p121)

13. Conic section equations
Given the coefficients of the general or standard equations for any of the conic sections (parabola, ellipse, circle or hyperbola) find the other form as well as all significant details (eg: centre, focii, vertices, lengths of major/minor axis, radius, latus rectum, roots, etc.) in addition to graphing the equation.

General Applications

14. 4 Bugs
4 bugs are placed at the four corners of a square screen. Each walks toward the next (clockwise) bug - directly. Plot their tracks.

15. Biorhythms
Given a persons date of birth, plot that person's biorhythms for a given period. The cycles are:

physical	23 days
emotional	28 days
intellectual	33 days

All cycles start at 0 at birth and follow a sine curve thereafter, with the above periods. Critical days are when the curves cross the x-axis. Triple critical days are the worse. (Input two persons birth dates and compare for compatibility.)
(Dyck 1984, p150)

Business Applications

16. Banking machine simulation
This can be complete involving graphics, and the ability to deposit, withdraw, bill payment, transfer, etc. (files may or may not be used). Can also include dates of last update so that interest can be added to accounts (on daily or monthly basis), and passbooks printed.

Games

17. Games
A variety of games can be created with the computer being the dealer (or a player) and the user is the (other) player, without using arrays:

a. Moon Lander
User controls vertical descent of a spacecraft landing on the moon. There is a single rocket thruster. Every 10 sec the user may enter a new throttle setting. Depending on what the user does the lander may crash or land successfully. The time, velocity, altitude, fuel and trust are continuously displayed. There are some complex formula to calculate for this.
(Dwyer 1984, p271)

b. 15
A version of tic-tac-toe without the graphics: players take turns taking the numbers 1-9 (once only) and first player to get 3 numbers to add to 15 wins.

c. Trio
Another version of tic-tac-toe. Difference is there are only 3 markers for each player and so after the first 3 moves, an existing marker is moved. The object of game is the same as in regular tic-tac-toe.

d. Nim
In its most general form there are n heaps, each containing a random number of objects. On each turn a player can select any number of objects from 1 to k heaps ($k < n$). The player to take the last object loses. With restrictions arrays are not necessary.
(Turing 1992, p304)

e. Black Jack
Can be simple or complex, complete with graphics, splitting, doubling down, etc. The cards used are assumed to consist of an infinite number of decks (therefore the use of arrays can be avoided) and any card is equally likely to be dealt at any time.
(Tamburin 1993, p5)

f. Greed
2 players take turns rolling a pair of dice. Each player can roll as many times as he wishes and their turn total is added to a grand total. The first player to get to x points wins. If a player rolls double 6's their turn total is 0 (and their turn is over), if a player rolls double 1's then their turn total and grand total is 0 (and their turn is over), any other double doubles their turn total.

g. Ones
Similar to Greed. Using dice, 1-1 -> reset total to 0, 1-x -> turn score = 0 (total is unaffected). x-x > turn score is 4x. Winner is the one to get to 100 points first. Both players must have = # of turns. Ties of two scores over 100 go to second player.
(U of Waterloo CS 130, assn#4, 2000spring)

h. Craps
(Tamburin 1993, p27)

i. Roulette
(Tamburin 1993, p39)

j. Baccarat
Only two hands, the dealer and player. Two or three cards are given to each (third card is given based on set rules). The hand closest to 9 wins. Tens and face cards count 0.

(Tamburin 1993, p47)

k. Big Six Wheel
(Tamburin 1993, p73)

l. Red Dog
Try to guess if the value of a randomly selected card is between two other cards. (Payoff depends on the spread of the two cards.)
(Tamburin 1993, p87)

m. Yacht/Yahtzee
This is a dice game, involving 5 dice, based on the principles of poker. The complete game can be played or, probabilities of getting various combinations of dice can be simulated.
(Dyck 1984, p498, Frey 1970, p273)

18. Arithmetic croquette
Two players, both starting at 0, alternate taking turns with the goal of making their number greater than 100. Each move consists of adding 1..8 to your number, except you may not take x and 9-x (where x is your opponents last move). For example if your opponent just took 6, then you cannot take 6 or 3. The numbers 10, 20, ... 100 are called hoops.They must be crossed such that the distance between the hoop and your number is the same before and after. For example if you were at 28, you must take 4 (giving 28 and 32, each 2 from 30). Alternatively you may take a hoop in two steps: 28 - 30 - 32. While on the hoop if you can't take the correct number (because your opponent took it) then you must skip your turn and wait for the number. (You can ONLY skip a turn if on a hoop.) If you should cross a hoop incorrectly, you must then subtract your number, get back across the hoop and then cross it correctly. (You may not skip a turn, except if you are on a hoop.)
(Gershtein 1995)

19. Pong
The old simple version. A boll shoots down from the top of the screen (or side) at a random angle and speed and the player moves a paddle (long rectangle) allow the bottom (or opposite side) with the aim of having the ball bounce off. Count the number of balls "bounced".

20. Prisoner's dilemma
Create a round robin tournament of the iterated Prisoner's Dilemma using a different strategy from each student. The iterated Prisoner's Dilemma involves two players who can choose to cooperate or defect and the points they get are determined by a pay-off matrix.

		A	
		coop	defect
B	coop	(c,c)	(s,p)
	defect	(p,s)	(d,d)

Where the values of c, d, p and s can be fixed to various values (eg 2, 0, 4 and -1 respectively). The goal is to accumulate as many points as possible over the course of the tournament. (One strategy is Tit-for-Tat: it will cooperate on the first turn, then thereafter it will do whatever the opponent did on the previous turn.)
(Hofstadter 1985, p715)

H Recursion

This contains recursion problems, which **DO NOT** involve 1D arrays, strings, trees, searching, sorting, etc.

That is to say: there are many recursion problems in all of the following sections.

1. Factorials
   - n! =1 if n = 0
   - n! = n(n-1)! if n > 0

2. Combinations
   - C(n,r) = 1 if n=r or r = 0
   - C(n,r) = C(n-1,r) + C(n-1, r-1) if n,r>=1

3. Fibonacci sequences
   - F(n) = n (or 1) if n =1 or 2
   - F(n) = F(n-1) + F(n-2) if n > 2

4. Even / odd
   - x is even if x = 0 or x-1 is odd.
   - x is odd if it's not even!

5. GCD
   - GCD(m,n) = m if n = 0
   - GCD(m,n) = GCD(n, m mod n) if n not= 0

6. Powers
   - $x^n = x^{n-1} * x$ if n > 0
   - $x^n = 1$ if n = 0

   or
   - $x^n = (x^{n/2})^2$ if n is even
   - $x^n = (x^{(n-1)/2})^2 * x$ if n is odd

7. Multiplication
   - ab = a(b-1) + a if b > 0
   - ab = 0 if b = 0

8. CountUp
   if n = 1 then put n, else countUp(n-1) and put n

9. Sum / product of n input numbers
   if x = 1 then get x and result x, else get x and result x + (or *) sum (or product) of (n-1)

10. Largest / smallest of n input numbers
    if x = 1 then get x and result x, else get x and result largest (or smallest) of x or largest (or smallest) of (n-1)

11. Print x/y as the sum of distinct reciprocals
    (0 < x/y < 1) Define:
    - f(x,y) if x = 0 then nothing to print
      else find smallest n such that 1/n <= x/y, print 1/n, call f(xn-y), yn)

12. Print big numbers with commas
Example 251462051 becomes 251,462,051
Writewithcommas (n)
    if n < 1000
        print n
    else
        Writewithcommas (n/1000)
        put ","
        Write3Digits (n % 1000) (this prints leading zeros if necessary)

13. N as a sum of r integers
Find (and combinations) the number of ways to express n as the sum of r integers (n >= r)
eg. The number of ways to express 7 with 3 numbers is 4 (1=1+5, 1+2+4, 1+3+3, 2+2+3). the number of ways to express 19 with 8 numbers is 52.

14. Hermite Polynomials
$H_0(x) = 1$
$H_1(x) = 2x$
$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x)$

15. Legendre Polynomial
$P_0 = 1$
$P_1 = x$
$P_n = (2n-1)/n*P_{n-1} - (n-1)/n*P_{n-2}$

16. Strange functions
Q(n) = 1 if n =1 or 2
Q(n) = Q(n - Q(n-1)) + Q(n - Q(n-2) if n > 2

F(n) = n - M(F(n-1)) if n > 0
M(n) = n - F(M(n-1)) if n > 0
F(n) = 1 if n = 0
M(n) = 0 if n = 0
(Hofstadter 1979, p137)

17. Recursive formula
a(n) = 1 n <=2
a(n) = a(a(n-1)) + a (n - a(n-1)) n > 2

18. Ackerman's function
A(0,n) = n+1
A(m,0) = A(m-1,1)
A(m,n) = A(m-1,A(m,n-1))
(Try this for m=0...3 and n = 0...5)

General Applications

19. Tower's of Hanoi
To Move(n=disks, s=source peg, t=target peg, i=intermediate peg)
    Move (n-1, s, i, t) if n > 1
    move nth disk from s to t
    Move (n-1, i, t, s) if n > 1

20. Visualizing Recursion
Print the local variables and parameters. Display results on separate line for each call and add indentation. Goal is to make output an aid to understanding. (Useful for simple recursion as well as the more complex recursive sorting routines.)

21. Draw a ruler
Make a ruler with the marks on it for ½, 1/4, 1/8, etc.

```
Ruler(l, r, height)
    if height > 0
        m = (l + r)/2
        drawmark (m, height)
        Ruler (l, m, height - 1)
        Ruler (m, r, height - 1)
```

(Sedgewick 1988, p54)

Fractals

22. Cantor set
Draw the Cantor set: draw a line in white then erase the middle third of the line (draw in black). Recursively do that to the first and last thirds of the line, for a given number of levels.

23. Cantor cheese
Draw a circle, then two others inside, recursively.

```
if level > 1
    draw circle with given x,y and r
    cheese (x-r/2, y, r/2, level-1)
    cheese (x+r/2, y, r/2, level-1)
```

(Algorithm 4.1, p12)

24. Sierpinski's triangle (and other similar fractals)

a. given 3 corner points and a level:

```
if level > 1
    compute the 3 midpoints
    call Sierpinski again 3 times using the 3 new triangles (omit the central one)
else
    plot the triangle
```

b. a box fractal
(Devaney 1990, p132)

c. sierpinski's carpet
(Pietgen 1991a, p94)

25. Square fractal
Given a procedure which draws a square centered at x,y of size 2r, then:

```
Fractal (x,y,r)
    if r > 0
        Fractal (x-r, y+r, r div 2)
        Fractal (x-r, y-r, r div 2)
        Fractal (x+r, y+r, r div 2)
        Fractal (x+r, y-r, r div 2)
        Square (x, r, r)
```

(Sedgewick 1988, p59)

I Classes

These problems include;

simple classes
composition of classes
inheritance

If you are NOT teaching OOP these provide a great source of function and procedure ideas.

1. Dates
   All problems dealing with dates can be dealt with using a date structure (day, month year). (Editing dates, calendar generation, adding a number of days to a date, subtracting dates, date conversion to/from Julian, etc.)

2. Time
   Time can be represented by a structure with hour, minute and second. Write a clock display program: digital or analog!

3. Turtle
   Create a series of procedures which mimic turtle drawing (of LOGO fame): forward, back, right, left, pendown, penup, home, clear, etc. and use these to:
   a. draw letters of various sizes
   b. complete pictures
   c. create a logo-type user interface. The program acts as an interpreter, reading a command at a time and acting on upon it.
   d. draw fractal shapes (Koch's curve, or Hilbert's curve) using these turtle graphics routines. (Peitgen 1991b, p34)

4. Radicals
   A radical of the form x*sqrt(y) can be represented with a structure: make a complete manipulation package (+, -, *, simplify, etc.)

5. Fractions
   A fraction can be represented by 2 integers. Include get fraction, print fraction as a mixed number, reduce fraction,+, -, *, /, display as part of a circle, convert to decimal, etc.

6. Monomials
   Consists of a coeff and exp. Get, put, and do arithmetic with these.

7. Complex number package
   A complex number can be represented by 2 reals. Include: get complex number, print, +, -, *, conjugate, magnitude, rectangular to polar conversions, powers, etc.

8. Real numbers
   Represent real numbers with 2 integers, left and right, representing the digits to the left and right of the decimal point respectively. The program should be able to read, write, +, -, *, etc. such numbers.

9. Complex numbers
   Complex numbers can be represented by two reals: make a complete manipulation package

10. Vectors

Two or three dimensions. All the standard operations.

11. Linear / quadratic equations
the coefficients of the equations are the fields. They can be plotted, as well as added and subtracted. Roots found. Points of intersection, etc.

12. Rectangular coordinates
Given a series of points various analytic geometry things can be done:
a. find the distance between points
b. plot the points

13. Triangles / rectangles / geometric shapes of many kinds
Standard I/O as well as properties such as perimeter, area, volume.

14. Find remaining parts of a triangle
Given one of: ASA, SAS, AAS, SSA or SSS, find the other angles and sides (need functions for law of sines and cosines). Note: in the case of SSA, if (cos(A)<=0 and S1<=S2) or (S1<S2sin(A)) then there is no solution.
(Some Common..., p53)

15. Fractal mountains: Midpoint Displacement Method
Given a triangle and a level

```
if level > 1
        calculate the midpoints of the sides
        displace the midpoints by a random amount
        call itself with the 4 new triangles
else
        draw the triangle
```

(Peterson 1988, p124)

Composition of classes

16. Point / lines / circles, etc.

17. Dice / pair of dice / dominoes / games
Beside domino games, there are also games such as craps and other dice games.

18. Card / Deck / Hand / Game
No end of possibilities here! (Eg. Evaluate Poker hands, determine which cards to throw, compete with a real player.)

19. Monomials / polynomials (linear/quadratic), etc.

Inheritance

20. Shapes
Shape is base abstract class. From it derive 2D shapes and 3D shapes. From those derive rectangles, circles, boxes and spheres, etc.

21. Shapes again
Like above but with different hierarchy. (Eg shape to line, circle and triangle. These to rectangle, sphere, pyramid, and so on.)

22. Shapes yet again
Shape to rect/oval /triangle to rect to square, oval to circle, circle to face, face to happy/sad, happy to rabbit.
(Hume 2000, p 435)

23. Employees
From the base employee class, salary, commission, or hourly working employees can be derived. Management employees can be derived from the salary employees, etc.

24. Tetris shapes
Base class is a tetrisblock which has a color and location. From that are lineblock (with orientation), squareblock and offsetblocks (abstract). Under the offset blocks you can have the 2and2 and 3and1 blocks (they have both offset and orientation). Use this series of classes to simply draw the shapes at various locations or make the complete game!

25. Exam Creation
Create a simple program to give exams. An exam is a list of questions. There are three varieties of questions: true / false, multiple-choice, short answer. Each question may have a weight -- the number of points it is worth. Your program will create a small exam, administer it by getting the user to answer the questions, and reports a percentage score. Your program should contain the following classes:
Question: An abstract class for the elements common to all kinds of questions. It should contain: a protected instance variable weight for the weight of the question, plus public get and set methods for this variable, a protected instance variable text for the text of the question, plus a public get method for this variable. The text of the question will be set by constructors and never changed, so you don't need a set method for it, an abstract method ask, which will ask the question, read the user's answer, and return a boolean result indicating whether the user answered correctly. It should also print a message to the user, telling them whether the answer was correct (and if it was incorrect, telling them what the correct answer was).
TFquestion: A subclass of Question, for true/false questions. It should contain: a concrete ask method. The legal answers are "t" or "T" for true, and "f" or "F" for false. If the user answers any string other than these four, ask the question over again. a private instance variable to record the correct answer (true or false) a constructor which creates a true/false question, given the text of the question, the correct answer, and a weight
ShortAnswerQuestion: A subclass of Question, for short-answer questions. For the purposes of this assignment, a short-answer question has only one possible answer; no judgement is needed in comparing answers. This class should contain: a concrete ask method. When comparing the user's answer to the correct answer, user equalsIgnoreCase, so that differences in case don't matter. a private instance variable to record the correct answer (a String) a constructor which creates a short answer question, given the text of the question, the correct answer, and a weight
MultChoiceQuestion: A subclass of Question, for multiple choice questions. A multiple choice question must have at least 2 choices. This class should contain a concrete ask method. It should print the question, followed by the choices, identified by the letters 'a', 'b', 'c', etc. private instance variables to record the choices (an array of Strings) and the correct answer (an int -- the index of the correct answer in the array of choices) a constructor which creates a multiple choice question, given the text of the question, the array of choices, the correct answer, and a weight
Exam: A class for a collection of question to be given as one exam. The exam starts out empty (no questions), and provides a method for adding questions to the exam. This class should contain: An array of questions. Later in the course, we will study ways to avoid the problem of a limit on the number of questions. For now, just create the array with size 100. A count of the number of questions currently in the array. A constructor (no arguments) to initialize the array and the count --

or you can put initializations in their declarations and just use the default constructor. a method addQuestion which takes a question (any kind) as an argument and adds it to the exama method giveExam which gives the exam to the user. It should ask each question in turn and keep track of the user's score. It should return an integer from 0 to 100, which is the user's score as a percent, rounded to the nearest integer.
(QueensU, 2001f, CISC124, Assn1)

J 1D Arrays

1. Basic array operations
   a. read / write
   b. print in reverse
   c. find total / average
   d. initialize array to all zeros, to (1,0,0,0,...), to (1,0,1,0,1,0,...) or to any pattern or sequence (arithmetic/geometric/Fibonacci)

2. Determine if a list is in sorted order

3. Add / remove elements
   a. given a value and a position insert the value after the given position (or remove the value at the given position)
   b. add a value at the end (or first) of an array (or remove the last (or first) element)

4. Drop duplicates
   a. from a sorted list
   b. from any array or string such as “Mississippi”

5. Compare 2 arrays
   Given 2 arrays, A and B, return 1 if A>B, 0 if A=B and -1 if A<B. A > B if A has more elements than B or if A has the largest non-equal element, starting from the left.

6. Reverse the elements of an array “in-place”

7. Concatenate 2 lists

8. Merge two sorted lists

9. Longest run
   Given a sorted array, find the length and starting location of the longest “run” in the array. A “run” is a sequence of numbers separated by 1, eg: 3,4,5.

10. Count frequencies
    a. of a number x in an array
    b. of random numbers (eg. a number of dice) and display the results graphically
    c. of marks in a set of ranges

11. Find min / max
    a. find the min/max of an array (or their positions)
    b. improve normal method by use of a “sentinel”
    c. Can be done recursively (see the following idea)
    (Bentley 1986, p173)

12. Sum (or product) of a series of numbers
    Given an array x, and logical size n,

```
if n=1
        return x_1
else
        return x_n + Sum (x, n-1)
```

13. Print in reverse
Given an array x with logical size n

```
if n > 1
        print x_n
        Print_Reverse (x, n-1)
```

14. Pivot
Given an array A and a value x, create a second array with all the values of A < x, followed by all the values in A = x and finally all the values of A > x.

15. Max distance
Given two arrays, x and y, both the same length and both with a decreasing sequence of numbers find the Max distance between them. The max distance is defined as D(x,y) = max {$d(x_i, y_j)$ for all i and j} where $d(x_i, y_j)$ is j- i if j>=i and $y_j >= x_i$, or 0 otherwise.

```
                    i
Example:    x = 8 8 4 4 4 3 3 3 1
            y = 9 9 8 8 6 5 5 4 3
                              j
```

Max Distance is 5.
(CCC 96)

16. Rotate (cycle) elements
Given an array and a number x, "rotate" the elements x positions to the left, eg: if the array was 1,2,3,4,5 and x=3, then the array would become 4,5,1,2,3. Methods:

a. copy the first x elements to a temp array, move the remaining elements x positions to the left and copy the temp array back to the last positions.
b. think of the rotation as swapping 2 segments so that the vector AB becomes BA (A is the first x elements). Depending on which is shorter A or B, split the other up so that one part is the same size as the first, ie: AB_LB_R such that A and B_R are the same size or A_LA_RB such that A_L and B are the same size. Swap the outside segments yielding B_RB_LA or BA_RA_L. In both cases one section is where it should be (A or B). Repeat the process on the other two segments.
c. let A be the first x positions of the array and B be the rest. Then reverse in-place A, then B then the entire thing: $(A^RB^R)^R = BA$

(Bentley 1986, p13)

Math Applications

17. Partial sums
Given an array of numbers, replace each number with the sum of all numbers up to and including that number.

18. Factorial Sum List
Given a #, calculate the sum of the factorials of its digits. Repeat with the new number. Stop when a repetition occurs. Eg: 25 -> 122 -> 5 -> 120 -> 4 -> 24 -> 26 -> 722 -> 5044 -> 169 -> 3636001 -> 1454 -> 169.

19. N-th order Fibonacci
Generate the nth order Fibonacci sequence given n and the first n numbers. The "next" term in the sequence is the sum of the previous n terms. Eg. the 4th order Fibonacci sequence starting with 5,2,6,10 is: 5, 2, 6, 10, 23, 41, 80, 154, 298, ...

20. Standard deviation
= sqrt(SUM(x_i - avg)^2/(n-1))
or = sqrt[(SUMx_i^2 - (SUMx_i)^2/n)/(n-1)]
(Dyck 1984, p366)

21. Weighted average
Find the weighted average of a series of numbers, given the weighting factors for each number
ie. given x_1, x_2, ..., x_n and f_1, f_2, ..., f_n, the weighted average is $f_1x_1 + f_2x_2 + ... + f_nx_n$
(where $0 <= f_i < 1$ and $f_1 + f_2 + ... + f_n = 1$)

22. Normalize a vector
Reduce all the elements to the range 0 to 1, with the smallest being 0 and the largest being 1. (For each element: x = (x-a)/(b -a) with a being the smallest element and b the largest.)

23. Basic vector manipulation
+, -, dot product, scalar multiplication, magnitude and angle between vectors ($\cos^{-1}$(uv/(|u||v|)

24. Pascal's triangle
Generate each line in the triangle from the previous

25. Bell numbers
Start with a 1 in row 1. For all the other rows, bring down the last number in the previous row to the beginning of the current row. Then each succeeding number in the row is obtained by adding the previous number and the number directly above it. The last number in each row is a Bell number.

```
1
1 2
2 3 5
5 7 10 15
...
```

The Bell numbers are: 1, 2, 5, 15, ...

26. Parkside's triangle
Given the size and seed, print the resulting triangle (you figure out the rules)

```
size = 6, seed = 1
1 2 4 7 2 7
  3 5 8 3 8
    6 9 4 9
      1 5 1
        6 2
          3
```

```
size = 5, seed = 3
3 4 6 9 4
  5 7 1 5
    8 2 6
      3 7
        8
```

```
size = 7, seed = 9
9 1 3 6 1 6 3
  2 4 7 2 7 4
    5 8 3 8 5
      9 4 9 6
        5 1 7
          2 8
            9
```

27. Max sum in a contiguous subvector
Find the maximum sum in a contiguous subvector of a given vector.
a. brute force (try every combination: 3 nested loops)
b. using partial sums (2 nested loops)

c. recursion (divide and conquer)
(Algorithm 3.4, p21, Bentley 1986, p69)

28. Patterns in random sequences
Generate x random digits (0..9) and test if there is a given sequence in them.

29. 9-digit perfect squares using all 9 digits
Find all 9 digits numbers that are perfect squares and uses each of the digits 1 to 9 only once. (Since the sum of its digits is 45, it implies that it is divisible by 9, thus the root is divisible by 3, thus test the numbers 11112 to 31425 in steps of 3. Also each square is 6r+9 greater than the previous. Use an array to test if each digit is used only once.)
(Wiesenberg1987, p178)

30. Primes and factors
a. find all primes up to x and store them in an array, then use this array to find all factors of a given number.
b. given an x, find PI(x) = number of primes <= x
(Giblin 1993, p40)
c. given an x, find the number of twin primes <= x
d. given a k and x, find the number of primes that differ by k that are <= x.

31. Factoring
a. for a given n, calculate Ù(n) = number of prime factors of n (eg 200 = 2*2*2*5*5, therefore Ù(200) = 5, also Ù(1) = 0)
b. for a given n, calculate ù(n) = number of distinct prime factors of n (eg ù(200) = 2)

32. Nasty numbers
Determine if a number is nasty. A nasty number has at least one pair of factors such that the difference of the pair = the sum of another pair. For example 6 is nasty because 6-1 = 2+3, 24 is nasty because 12-2 = 4+6
(CCC 1997)

33. Exponential prime power (EPP) representation
because every number can be written as the product of prime powers, eg: n = $2^{p2}*3^{p3}*5^{p5}*...*r^{pr}*r1^{0}*r2^{0}$ ...
And because the primes are known (at least in theory), the number n can be represented by the list (p2, p3, p5, .. pr), some of which may be zero. Applications of this include fast multiplication and finding GCD's.
(Nissen 1995)

34. Euler's function (or totient function)
a. for a given n, calculate Ö(n) = number of integers x such that 1<=x<=n and GCD(x,n)=1.
i. by brute force method based on definition
ii. start Ö at n and multiply by (p-1)/p once for every p (prime) dividing n.
b. find all n such that Ö(n) = Ö(n+1) or Ö(n) = Ö(n+3) for n < 10000.
c. iterate Euler's function: that is given an n find Ö(n), Ö(Ö(n)), etc. The value should eventually become 2.
(Giblin 1993, p116)

35. Decimal expansion of a given fraction (Repeating decimals)
Given a fraction n/d find it's equivalent decimal expansion. If the decimal repeats, enclose it in brackets as in 43/14 = 3.0(714285)

36. Order of a mod n and length of decimal period
    a. calculate the order of a mod n. If the GCD(a,n) = 1 then the smallest k>0 such that a^k=1 mod n is called the order of a mod n and is written k = $ord_n a$.
    b. compute the length of the decimal period of 1/m and verify that if GCD(m, 10) = 1 then the order of 10 mod m = length of the decimal period.

    (Giblin 1993, p120&131)

37. Primitive root mod n
    a. given numbers a and n determine if a is a primitive root mod n. A number a with GCD(a,n) = 1 and the $ord_n a$ = Ö(n) is called a primitive root mod n.
    b. find all primes p < 100 for which the $ord_p 10$ = p-1 (ie. 10 is a primitive root mod p)

    (Giblin 1993, p121&131)

38. Divisor function
    a. find the value of d(n) = number of x such that 1<=x<=n and x divides into n. Use the definition to calculate d(n).
    b. find d(n) using the following:
    Let $n = p_1^{n1}p_2^{n2}...p_k^{nk}$ then $d(n) = (n_1+1)(n_2+1)...(n_k+1)$ and d(1) = 1.

    (Giblin 1993, p158)

39. Sum of divisors, amicable and perfect numbers
    a. find the sum of the divisors of n using the brute force method from the definition: ó(n) = sum of all x such that 1<=x<=n and x divides into n.
    b. find ó(n) using the following: for each prime which divides n, calculate the total $1+p+p^2+p^3+...$ as long as the power of p divide into n.
    c. define s(n) = ó(n) - n. Then iterate s(n) for a given n (ie: calculate s(n), s(s(n)), etc.) Do all sequences end with 1? or are some periodic?
    d. find amicable (friendly) pairs of numbers m and n. m and n are amicable pairs if m not= n and ó(m) = ó(n) = m+n (or s(n)=m and s(m)=n).
    e. find all perfect numbers less than x. A number n is perfect if ó(n)-n = n (or ó(n) = 2n). If ó(n)<2n then n is deficient and if ó(n)>2n it is abundant.

    (Giblin 1993, p164)

40. Inverses mod n (ie: find x such that ax mod n=1 where GCD(a,n)=1)
    a. the algorithm is:

```
g(0) = n, g(1) = a
v(0)=0, v(1) = 1
i = 1
while g(i) not= 0
        y = g(i-1) div g(i)
        g(i+1) = g(i-1) - y*g(i)
        v(i+1) = v(i-1) - y*v(i)
        i = i +1
x = v(i-1)
if x < 0
        x = x + n
```

    b. Or use the following equation:
    $x = a^{Ö(n)-1} \bmod n$

    (Denning 1982, p44)

41. Sieve of Eratosthenes
    Find primes using this method:

set all numbers in array P (2 to n) to 0
set i to 2
while i < n
print i as prime
for all j, such that i*j <= n, set $P_{i*j} = 1$
find next i such that $P_i = 0$

(Kochan 1983, p98, Giblin 1993, p55)

42. Pythagorean Triples

Find all Pythagorean triples (eg 3,4,5 or 5, 12, 13), eliminating duplicates and multiples. (Eg 6,8,10 doesn't qualify)

43. Good Sets (the Erdos Problem)

A good set is a set of integers in the range 1 to n such that the sum of no 2 is a perfect square. (This includes a # and itself.) Find large good sets by following the method: start C = {1,2,3,...,n}. Remove all number which are ½ of a perfect square. Then for each randomly selected x in C (put in the good set) remove all #'s y in C such that x+y = perfect square.

(SMSU pp#43)

44. Lucky numbers

Find the lucky numbers by the following sieve method:

- list all the positive integers
- other than 1, the first number is 2, so cross out every second number starting the count at 1.
- other than 1, the first number left is 3, so cross out every third number starting the count at 1.
- continue in this fashion.
- the numbers left are "lucky"!

(Giblin 1993, p67)

45. Linear interpolation

Given two arrays x and y, representing the values of a function (x is in ascending order) and a point a (within the range of x values), find the corresponding value of y. If a is one of the x's, then the answer is the corresponding y, but if not, then the answer is:

$y_1 + ((a - x_1)/(x_2 - x_1))(y_2 - y_1)$

where $x_1 < a < x_2$.

46. Least squares best-fit line

Given a set of data points x_i and y_i (i=1..n), find the equation of the line that best fits the data. The equation is: $y = mx + b$ (find m and b)

<x> = SUM(x)/n
<y> = SUM(y)/n
<xy> = SUM(xy)/n
$<x^2>$ = SUM(x^2)/n

m=(<xy>-<x><y>)/(<x2>-<x><y>)
b = <y> - m<x>

(Borse 1985, p463)

47. Continued fractions (converting decimal to fraction)

To convert a given decimal x_0 into a continued fraction: $x_0 = r_0 + 1/(r_1 + 1/(r_2 + ... (1/r_n))))$
$r_i = INT(x_i)$ and $x_{i+1} = 1/(x_i - r_i)$, stop when $r_i = x_i$.
To simplify this to an ordinary fraction, set

num = r_n
den = 1
for each i = n-1 to 0
a = num
num = r_i*num + den
den = a

The final num/den is the normal fraction.
(Dwyer 1984, p135)

48. Continued fractions (converting normal to continued and reverse)
Given a normal fraction a/b convert it into a continued fraction:
eg: 27/8 = 3 + 3/8 = 3 + 1/(8/3) = 3 + 1/(2 + 2/3) = ...
(Brown 1988, p315)

49. Factor a trinomial
Given a, b and c in the expression $ax^2 + bx + c$, find the numbers m, n, p, s, t such that the expression = m (nx + p)(sx + t)
eg: $6x^2 + 34x - 12 = 2(3x - 1)(x + 6)$

50. Polynomial manipulation
Store coefficients in array with indices: 0 to n
input, output, +, -, mono multiplication, full multiplication, evaluation (including Horner's Rule:$((a_3x+a_2)x+a_1)x+a_0$), derivatives, integrals, graphing, finding roots (use Newton's method), determine the max number of positive real roots (= # of sign changes of the coefficients) and the max number of negative real roots (= # of sign changes in the coefficients when x is replaced by -x) Descartes Law of Signs.

51. Big number manipulation
Represent a very large integer by storing several digits in each element of the array (may include the sign and or length in the first element)
input, output, +, -, 'scalar' multiplication (a normal number with a big number), full multiplication, etc.

52. Coordinates and polygons
Store the coordinates of the vertices of a polygon in an array of structures and then graph or find the area of the polygon, etc.
Area = ½ SUM$[(x_{i+1}-x_i)(y_{i+1}+y_i)]$ + $(x_1-x_n)(y_n+y_1)/2$
Perimeter = SUM{sqrt$[(x_{i+1}-x_i)^2 +(y_{i+1}-y_i)^2]$ + sqrt$[(x_1-x_n)^2 +(y_1-y_n)^2]$}
(in both cases SUM goes from 1 to n-1)
The coordinates of the vertices must be in order around the polygon.
(Dyck 1984, p297)

53. Graphing polygons
Create structures for points and vectors and use them to create, draw and manipulate (ie. translate, rotate, scale, reflect, projections, hidden line removal, etc.) various shapes in 2D or 3D
(Bielig 1990)

General Applications

54. Babbling Brooks (Insertion and deletion in an array)
Input is a series of numbers: the first number, n, is the number of brooks at the top of the mountain, then is a series of n numbers representing the flow for each. Then 99 indicates a split, followed by

the number of brook to split and % flow in the left. An 88 represents a join and then the number of the stream to join with the one to its right. Stop when 77 entered. Print the flows in each stream at bottom of mountain. For example: Input of 3, 10, 20, 30, 99, 1, 50, 88, 3, 88, 2, 77 gives:

top: 10 20 30
5 5 20 30
5 5 50
bottom: 5 55
(CCC 2000)

55. Find peaks
Given a series of numbers, (representing daily air pollution counts) find the peaks (a # number which is greater than the number before and after) and output the day on which it occurred. Also find the largest peak.

56. Mark calculations - drop lowest
Given a series of assignments and test marks (raw score, marked out of totals and weightings) calculate final mark, including the lowest assignment or test mark that gives the best average.

57. Traffic survey
A detector in a road sends a signal every second. 1=no vehicle, 2=vehicle, 0=end of survey. Given a series of these data elements, calculate the length of the survey, the number of vehicles, longest interval with no vehicles, average # of vehicles per minute, etc.
(Findlay 1985, p70)

58. Birthday pairs in a crowd: Birthday Paradox
Generate a series of random numbers in the range 1-365 and store in an array. Determine the number of duplicates. Repeat many times so that you can determine:
a. the number of birthday pairs given a crowd of size x.
b. the smallest crowd size needed to give a >50% chance of having at least one birthday pair. (This is the Birthday Paradox: the answer is 23.)
c. Generalized problem: how many people have to gather to give a more than x% probability that y-people share the same birthday. (If x = 50 and y = 3 ans= 88, y = 4 ans = 187, y = 5 ans = 313.)

59. Simple time table problem
Input a series of data containing: teacher, period, and room (in random order) then print a report by period (ie. all period 1 classes, etc.)
(Hume 1990, p73)

60. Treasure hunt
Given an array of integers and a starting location, use the value at that location as the position of the next location and continue until you go off the array, in which case you have got to the treasure and won, or you repeat a location. For example (5, 3, 8, 2, 4, 7) and a starting location of 1 leads to the sequence: 1, 5, 4, 2, 3, 8, treasure!

61. Floyd's algorithm: Get a random permutation of M #s from N #s
Given M and N with M <= N, find a permutation of M elements from 1..N. Store the permutation in S.
Algorithm:

```
for j : n-m+1 to n
    t = random(1,j)
    if t in s
```

```
                    insert j after t in s
                else
                    prefix t to s
```

(Bentley, 1988, p142)

62. Transposition and derangements

Given a list, a random transposition is obtained by getting two random numbers, i and j and swapping the ith an jth elements in the list. A permuted list is "deranged" if no element remains in its original position. Determine the average # of random transpositions needed to get a deranged list.
(Brainerd 1982, p301)

63. Next permutation

Generate the "next" permutation of the sequence 1,2,3,...n, given the current sequence. The algorithm is:

- find the longest decreasing subsequence at the end of the current subsequence (let this subsequence start at k+1. If k = 0, all elements are in decreasing order and all permutations have been generated.)
- convert this subsequence into an increasing one (ie. reverse in place)
- find the smallest element in this subsequence that is greater than a(k) and call it a(i)
- swap a(k) and a(i).

(Ammeraal 1992, p85)

64. Permutation inverses and inverse table

Given a permutation of the first n integers, find its inverse and inverse table. The inverse tells where each number is:

if the permutation is: 591826473
its inverse is: 359716842
because 1 is in the 3rd position, 2 is in the 5th position etc.

The inverse table tells how many numbers to the left of a number are greater than that number:

its inverse table is: 236402210
because there are 2 numbers bigger than 1 (to its left), 3 numbers bigger than 2 (to its left), etc.

(Brown 1983, p200)

65. Hoare's algorithm: Select the kth smallest element from an array

The idea is to permute the elements of x so that

x[1..k-1] <= x[k] <= x[k+1..n]

Algorithm

```
l = 1
u = n
while l < u
    swap (x[l], x[ random(l,u)])
    m = l
    for i = l+1 to u
        if x[i] < x[l]
            m = m + 1
            swap (x[m], x[i])
    swap (x[m], x[l])
    if k <= m
        u = m - 1
    else
        l = m + 1
```

(Bentley 1988, p159)

66. Stuffing mailboxes
given the number of boxes and surveys, stuff the mailboxes as follows: Put survey A in all mailboxes. Put survey b in every second mailbox starting at the second box. Put survey C in every third mailbox starting at the third box, etc. when done print the box 3 with most surveys, the number of surveys in box x, all the boxes with only survey y, all the boxes with exactly w surveys, total # of surveys stuffed.

67. Loading aircraft
Given a series of separate items of different weights, find how many trips are necessary to carry them all and the total payload of each trip, given the maximum payload of the aircraft. (How will you handle an item whose weight is greater than the maximum payload?)

68. Simplified knapsack problem
Given the sum s and a list of positive integers, find the subset whose elements sum to s, if possible.
(Ammeraal 1992, p92)

69. Knapsack problem
Given a capacity of the knapsack M, and a set of many items of N different sizes and values, find the combination of items which maximizes the total value of items in the knapsack.
(Sedgewick 1988, p596)

70. Knapsack problem variant (Dynamic Programming)
Given an array of numbers (a) find the combination of those numbers which add to a given sum. That is, say whether it is possible to get that sum and what is the minimum number of numbers were used to get the sum. Eg you have the numbers 1, 3, 4 and the sum is 13, the minimum number is 4 (3+3+3+4) Note: you may use more than one of any number in the array.
The solution is: Let F(x) be an array which contains the minimum number of numbers necessary to reach a sum of x. Define F(0) = 0. The array is created using:
F(x) = 1 + minimum of all the F(x - a[i]) (i goes from 0 to size of a)
For example if the sum was 10 and a was: 5, 9, 2

F(0) = 0	
F(1) = ?	(no solution: set to -1)
F(2) = 1	(2)
F(3) = ?	
F(4) = 2	(2+2)
F(5) = 1	(5)
F(6) = 3	(2+2+2)
F(7) = 2	(5+2)
F(8) = 4	(2+2+2+2)
F(9) = 1	(9)
F(10) = 2	(5+5)

the last F(10), for example was obtained from the minimum of
F(10-5) = F(5) = 1
F(10-9) = F(1) = ?
F(10-2) = F(8) = 4
F(5) was the minimum so F(10) = 1 + F(5) = 2
Extension: state the numbers used!
(CCC 2000)

71. Facebender / morphing

Given the digitized version of the face (made from a photograph, 186 key coordinates) and the "normal" face (on file) and an exaggeration factor, draw the caricature of that face. If the other face is not the "normal" face, morphing can be done.
(Dewdney 1988, p 960)

Fractals and Cellular Automata

72. Line automata
Given k (number of states allowed per cell), r (the radius of neighbourhoods used to compute the next state), a set of rules (encoded as a single number) and an initial population, generate the line automata for as many generations as desired.
(Dewdney 1988, p140)

73. Orbits and repetition length
Given a function and an initial point, calculate the orbit of the function and stop after either x times or after the orbit repeats. Find the repetition length.

74. Sandpiles: self-organized criticality
Both 1D and 2D simulations are possible. Either the pile is built up one grain of sand at a time, or a wet pile is allowed to dry and "relax". The idea is that in all cases the height difference between 2 points cannot be greater than some fixed critical value. If they are the sand will tumble down.
(Bak 1988)

Games

75. Peg game
There are 6 pegs and a hole arranged as:

A B C _ D E F

To move, take a peg and jump an adjacent peg into a hole and remove the jumped peg. The object is to end up with only one peg in the centre. Program should print a list of moves to show how to win.
(I believe the idea is to have a recursive WinningPosition function, if only one in centre return true, else for all possible moves, check if they result in a WinningPosition and if so print that move, otherwise if there are no possible moves, return false.)

76. Shuttle puzzle
Given a size n, the puzzle consists of n white pieces, a gap and then n black pieces. The object is to reverse the white and black pieces (ie so that all the black pieces are to the left of the gap). There are only 2 moves allowed:

move a piece into the gap
jump one piece over another of the opposite color into the gap (this can only be done in the direction the pieces need to go: ie. black pieces only jump to the left)

One possible algorithm: (recursion is not strictly necessary)

```
if game's not over
    if possible:    jump
    else:           move gap towards the centre
    repeat
```

77. Shuffle a deck of cards
a. simply generate 52 random numbers with no duplicates (a temp boolean array is easiest for this)

b. load the deck with the numbers 1...52 then for each element pick a random number and exchange its value with the current element's value

c. Perfect Interleave
Store the numbers 1..52 in an array. Split the array into two 26 element arrays and interleave the elements to reform a 52 element array (ie. take one from one list, the next from the other, etc.). Repeat many times.

d. Random Interleave
As in perfect interleave, except get a random # (1,2,or 3) and copy that many over from one list at a time.

78. Sports judging scores
Given a set of scores, find the total/average if the highest and lowest are dropped.

79. Bridge (the card game)
Deal a complete round of Bridge and display as in the newspaper (with N, E, S, and W). Evaluate each hand:

1pt for each Jack
2pt of each Queen
3pt for each King
4pt for each Ace
1pt for each suit with only 2 cards
2pt for each suit with only 1 card
3 pt for each suit with no cards

80. Cribbage
Given a hand of 4 cards plus the "start" card, evaluate the hand:

4 of a kind		12pts
3 of a kind		6
pair		2
groups totalling 15		2
run of 3	3	
run of 4	4	
run of 5	5	
flush of 4		4
flush of 5		5
Jack same suit as "start"		1

81. Bowling

a. maintain bowling scores for 4 players in 12 frames. Given 2 scores for 2 balls in each frame (sum <= 10) the score is their sum. However if the first score is 10 (a strike), then the score for the frame is 10 plus the score of the two balls in the next frame (if the first of those scores is also 10, then add the first score of the following frame). Finally if the score of the two balls is 10 (a spare) then the total score for the frame is 10 plus the score of the first ball from the next frame.
(Solow 1988, p493)

b. play a complete game
(Stephenson 1994, Gr 12 Proj.)

82. Simple pinball (Plinko on "the Price is Right")
There are x rows of pins on a slanted table with x+1 pockets on the bottom (this ratio need not necessarily hold). Set the position of the ball, $P = x/2 + 1$ (or have the user set it). As the ball falls, it will hit a pin on each row and $P = P \pm 0.5$ (if user sets position, or the x to x+1 ratio if not used,

check for out of bounds). Repeat many times, keep count of final positions and display as a bar graph. Could also include graphics to show pins and ball falling. Could give the final pockets values and play this as the game "Plinko".
(Dwyer 1984, p130)

83. Nim
In its most general form there are n heaps, each containing a random number of objects. On each turn a player can select any number of objects from 1 to k heaps (k < n). The player to take the last object loses.
(Turing 1992, p304)

84. Circles (a version of Nim)
A series of circles are drawn as:

```
   0
  000
 00000
0000000
```

Players alternate crossing out circles (they must be together and not separated by a crossed out circle and they must be all on one row). The player who takes the last circle looses!

85. Bulgarian solitaire
This is played with n stones divided into q piles. Each pile need not contain the same number of stones. After an initial placement, each round consists of taking 1 stone from each pile and use these stones to create a new pile. A "winning position" is one which is repeated on any subsequent play. A "losing position" is one which repeats an earlier (but not the previous) configuration.
(Brown 1988, p257)

86. Reversal game
Display the digits 0 to 9 in random order. Then ask the user to choose a number N, after which the computer will reverse the first N numbers in the sequence. The object of the game is to get all numbers in order in the fewest number of reversals.
(Carter 1989, p365)

87. Top Spin
The numbers 1 to 20 are arrange on an oval track in random order. 4 numbers on the top can be reversed (as a unit). The numbers can be moved around the track in either direction any number of places. Try to get all the numbers in ascending order.

88. Games:
a. Slot Machine
(Regan 1985)
b. Mastermind
c. Crazy Eights
d. Go-Fish
e. Dominos
f. Board games such as Trouble, Monopoly, etc.

89. Juniper Green
The game is played by two players who alternate moves. The first move of the game must be an even number. Each subsequent move must either be an integer multiple or integer divisor of the previous move. (Can't take a number twice.) The last player who is able to make a move is the winner. (You can use the numbers from 1 to 116 there abouts, omitting the primes over 50.)

(Scientific American March 1997, 118)

90. Mastermind

With a code of only 4 long, it is possible to guess the correct answer in 5 guesses or less. See. “Computer as Master Mind” by Donald Knuth in J. Rec. Math vol 9(1) 1976-7. Might be able to do it by maintaining an array of possible guesses and prune it ourselves without his complex logic, but I’m not sure... (Quite complex and may be a linked list exercise as well. Best done as a major project methinks.)

K Strings

1. Basic operations with 2 strings
   concatenate
   copy
   test for equality

2. Convert name forms
   Input a name in one form and convert to another:
   eg: John S. Mill to Mill, John S. (or Mill, J.S.)

3. Convert date forms
   Convert a date in the form “dd/mm/yy” (or “MMDDYY”) into “8 March 1965".

4. Convert time forms
   a. convert time from “hh:mm” into an integer representing the total number of minutes (or reverse)
   b. convert “hh:mm:ss” (24 hour clock) into more normal form such as “2:30 pm”.

5. Upper/Lower case change
   Change given text into all upper case or lower case or change the first letter of each word (or sentence) to upper and the rest to lower, etc.

6. Replace blanks
   Replace all sequences of blanks in a given string with a single blank

7. Pad string with blanks
   Pad a given string with blanks to make it x long.

8. Leading/trailing blanks
   Remove leading/trailing blanks from a given string

9. Justify string
   Given a string and a field width, justify (left, right or centre) the string in the field

10. Justify text
    Given multiple lines of text of various lengths, and a width, justify the text for that width (either left, right or full)

11. Newspaper columns
    Given some text, a total width and number of columns, format the output to fit in the required number of columns (left or full justified). Each column should have the same number of lines except the last.

12. Count character types
    Count the number of vowels, consonants, digits, white space characters, etc in some given text.

13. Count letters
    Count the single character frequencies

14. Split a line into separate words

15. Count the words in some text

16. Count words of each length
Count the number of words of each length in some text.

17. Average word length
Find the average word length in some text, or the average number of words per sentence.

18. Substring processing
    a. extract the substring given the string, start and length of the substring
    b. find the location of the start of a given substring in a given string
    c. remove a substring from a given string, given the location and length of the substring
    d. insert a new given string into another given string at a given location
    e. given a string, replace all occurrences (or just the first) of a given substring with another substring

19. Finding substrings
    a. brute-force method
    b. Boyer-Moore method (comparisons proceed right to left and with 2 tables it is a very efficient searching method)
(Dewdney 1989, p365, Sedgewick 1988, p286)

20. Letter removal
Given a phrase and a letter (or series of letters) remove all occurrences of that letter(s) in the phrase.

21. Change substrings of longer words
For example change all "or" to "our" in words like "color", but not in the word "or" itself.

22. I before E rule
Check a given word for adherence to the rule "i" before "e" except after "c".

General Applications

23. Palindromes
Determine if a string is a palindrome.

24. Palindromes (recursive algorithm)
A string is a palindrome if the first and last characters are equal and the remaining substring is a palindrome or the string is only one character.

25. Edit postal code
Given a postal code, edit it for proper placement of capital letters and digits.

26. Product code validation
eg: AX6BYU56UX6CV6BNT7NM 287430
1st part can contain only capital letters and 6 digits. 2nd part is all digits and = the product of the first 6 digits taken in groups of two from the left. Eg 65*66*67 = 287430.

27. Readability Index
Given some text, determine it's readability index = 0.4(w/s) + 12(l/w) - 16 where s = # sentences, w = # words and l = # syllables (= #vowels in a word, except 2 adjacent vowels = 1 vowel and if there is a vowel+consonant+"e" = 1 vowel and all words have at least one syllable.)

(SMSU pp#14)

28. Likeness sequence
Generate the “likeness” sequence.

1
11
21
1211
111221
312211
...

The idea is the first one is 1, the others describe the previous. for example the last line can be read, “three 1's, two 2's and one 1". The algorithm is simply to obtain the previous sequence then “describe” it.
(Algorithm 4.1, p11)

29. Pig latin generator
Take some text and change it into pig latin. A pig latin word is made by taking the first letter and putting it on the end and adding an “a”: for example “pig” becomes “igpa”.

30. Square word patterns
Given a word, print a square made from letter rotations of the word as in:

help
elph
lphe
phel

31. Hollow Squares
Given a string (eg “four”) print a hollow square as in:

```
fourf
r   o
u   u
o   r
fruof
```

32. Word diamonds
Given a word which starts and ends with the same letter a word diamond can be formed as in: “area”

```
  a                 a
 r r               e r
e   e             r   e
a     a  or       a     a
 r   r             e   r
  e e               r e
   a                 a
```

33. File compression (Run length encoding)
Replace repeated strings of characters with a single instance of the repeated character along with a count of the times the character was repeated. (An escape sequence consisting of an escape character, the count and the repeated character may be necessary if numbers themselves are in the text.)

34. Dynamic dictionary coding

Replace duplicate words with its position in a dictionary. If a word is not in the dictionary, it appears as is and is added to the end of the dictionary. (Do both encoding and decoding)

Example: the cat chased the rat while the dog chased the cat into the rat house
becomes: the cat chased 1 rat while 1 dog 3 1 2 into 1 4 house
(CCC 1997)

35. Cryptography

a. Transposition ciphers

Rearranging the letters of the plain text

i. Columnar Transposition

Given the number of columns, the plain text is written in matrix form, eg if the plaintext was "renaissance" and the number of columns = 4, then the matrix would be:

```
r e n a
i s s a
n c e
```

The ciphertext is obtained by taking off the columns in a given order (eg if the order was 2-4-1-3 then the cipher text would be: "escaarinnse"

ii. Fixed period

Given a number d, and the numbers 1..d in some permutation, then blocks of d characters from the plaintext are switched around according to the permutation. Eg: if d=4 and the permutation was (2, 4, 1, 3) then:

plaintext: rena issa nce
ciphertext: earn sais cne

(Denning 1982, p59-60)

b. Substitution ciphers

Each character of the plaintext is replaced with the corresponding character of a cipher alphabet.

i. Keyword Mixed alphabet

Given a keyword, the cipher alphabet is created by listing the letters of the word, omitting duplicates, followed by the remaining letters of the alphabet

ii. Shifted alphabets

Each letter of the plaintext is shifted right by k positions (mod the size of the alphabet)

$f(a) = (a + k) \bmod n$

(the Ceaser Cipher is when k=3)

iii. Shifted Alphabets based on multiplication

Same as the shifted, except:

$f(a) = a*k \bmod n$

iv. Affine Transformations

Given 2 keys, k1 and k2

$f(a) = (a*k1 + k2) \bmod n$

v. Vigenere Cipher

Given a key word, use each letter as the amount to shift the corresponding letter of the plaintext, eg if the keyword was "band":

plaintext: rena issa nce
band band ban
ciphertext: sead jsfd ocr

(Denning 1982, p62-74)

c. RSA cipher

(Denning 1982, p104)

36. Compare permuted strings
Compare 2 strings of equal length to see if the second string can be formed by permuting the characters of the first string.
(Hume 1990, p229)

37. Cycles of a Permutation
given a permutation of the first x letters, eg h,c,d,b,j,e,i,g,a,f express this as a series of cycles. Eg (a,h,g,i) (b,c,d) and (e,j,f)
(SMSU pp#39)

38. Well ordered words
Given a series of letters (eg "computer") and a length (eg 3), print all the well ordered words of the given length from those letters. (well ordered means in alpha order). Eg from the above example there are cem, cmr, ceo, cmt, mrt, rtu, ... for a total of 56.

39. Permutations
Print all the permutations of a given string. The "trick" to the solution of this is to pass to the procedure 2 strings, a front and back. Originally, front is empty and back contains the string to be permuted.

```
if length of back = 1
        print front + back
else
        for i = 1 to length of back
                Permute (front+ith letter of back, back without ith letter)
```

40. Combinations
Print all combinations of r characters from a given string of n characters. The "trick" to this is like that for permutations, a front and back string are used. Originally front is empty and back contains the string.

```
Comb(front, back, r)
        n = length of back
        if r = 0
                print front
        else if r=n
                print front + back
        else
                for i = 1 to n+1 - r
                        Comb( front + back(i), back(i+1..*), r-1)
```

41. Bit Patterns
Given n and k, print all bit patterns of length n with k 1's. Eg n= 6, k = 3
111000, 110100, 110010, 110001, 101100, ...
Idea is to start with all ones to the left. Move the last 1 to right if possible. If not move next last one to right once and move all ones at far right to join it. (Eg: after 011001, it would be 010110). Stop when all one's are as far right as possible.
(CCC 96)

42. Pick Lists
Given a list of words, determine the number of characters, starting from the left, that must be given to uniquely identify the words. Eg the answer is 5 if the words are: slink, slipper, slop, sloppy. The answer is 4 if the words are: additional, address, names, reports.

43. Significant digits
Given a number, (as a string), identify or count the number of significant digits.

44. Morse code
Convert given text into Morse code:

a(.-)	b(-...)	c(-.-.)	d(-..)	e(.)	f(..-.)	g(--.)
h(....)	i(..)	j (.---)	k(-.-)	l(.-..)	m(--)	n(-.)
o(---)	p(.--.)	q(--.-)	r(.-.)	s(...)	t(-)	u(..-)
v(...-)	w(.--)	x(-..-)	y(-.--)	z(--..)		
1(.----)	2(..---)	3(...--)	4(....-)	5(.....)	6(-....)	7(--...) 8(---..) 9(----.) 0(-----)

(Koffman 1994, p138)

45. LISP-like functions
a. CAR: return the first word of a given sentence
b. CDR: return the given sentence, without the first word.

46. LISP interpreter
Evaluate simple LISP expressions involving:

ADD (x1, x2, x3, ...)	sum the arguments
MULT (x1, x2, x3, ...)	multiply the arguments
CAR (list)	return first element in list
CDR (list)	return all but the first element in list
CONS a (list)	add a to the list

for example evaluate:

ADD (CAR (7 3)) (CAR (7 8))	= 14
CONS (CDR (1 2 3 4)) (CDR (5 6 7 8))	= ((2 3 4) 6 7 8)

note: brackets are critical!
(Brown 1988, p209)

47. Simple interpreter
Write a simple interpreter which will handle a program such as:

```
variable a
variable b
variable c
a = 3
input b
c = a * b
output c
done
```

Reserved words : variable, input, output and done. Only + * as operators. All token separated by a space. Variable name can be any number of characters. Error message include variable not declared, unknown operator and missing 2nd operand.

48. Machine language programming
Write a program which will run a program written in Simpletron Machine language (SML) an easy version of assembler.
(Deitel 1994, p328)

49. Determine if a string is alpha (lower/upper case), numeric, etc.
a. normal series of questions method

b. a better method is to load an array for each character and store the type as in type["a"] = lower case letter, type["B"] = upper case letter, etc. then use a simple table look up to determine the type.

(Bentley 1986, p82)

50. Parity checking

a. given a 7-bit binary number, attach an 8th parity bit to it so that the total number of 1 bits (including the parity bit) is even (or odd).

b. given an 8-bit binary number, determine if it the parity bit is correctly set (either for even or odd parity)

51. Pattern matching with wild cards

Given a dictionary of words, find all words in the dictionary that match a given pattern which may contain the following wild cards:

"?" matches any single character

"*" matches any 0 or more characters

For example "w?d*" matches "Wednesday" or "wedding" but not "weeding"

52. Distance between cities, given latitude and longitude

Given the latitude (N/S) and longitude (E/W) of a series of cities (eg: 41°49'N or 131°47'E), find those two cites which are the furthest apart. Convert the latitude and longitude into polar angles a and b

a = 90 - latitude (if N)

a = 90 + latitude (if S)

b = longitude (if W)

b = 360 - longitude (if E)

angle between the 2 points is theta,

$\cos(\text{theta}) = \cos(a_1)\cos(a_2) + \sin(a_1)\sin(a_2)\cos(b_2 - b_1)$

then distance = theta*r (where r = 3958.89 miles)

(Borse 1985, p283)

53. Multiple choice test marker

Given the solutions to x questions, and then a series of student names and their answers, create a report of student names and mark and class average. The format of the input can be: solutions on one line, each name and answers on a separate line. The solutions and answers can be in fixed locations or separated by an unknown number of spaces (in which case the student name must be only one word).

54. Voting with multiple choices

Each ballot (input line) has 1st, 2nd, 3rd,... choices on it. Each can have any number of choices >= 1. If a ballot has a non-existent candidate or votes twice for same candidate, reject it. Look at 1st choices, if a winner, done. If a tie, look to 2nd choices, etc.

(SMSU pp#40)

55. Genetic algorithms

(Can be used to solve a variety of problems, including mastermind, or finding the min/max of a function.)

This is based of biological evolution:

- set up an initial population of "solutions", created at random. The solutions are stored as strings for easy manipulation.
- evaluate each solution for "fitness" (In mastermind compare it to the true solution, or in min/max problem, evaluate the function using the solution.)

- create the next generation by taking the best solutions from the previous generation and "mating" them. ("Survival of the fittest"). Mating is preformed by the process of crossover: the offspring is made from the front part of one parent and the back part of the other parent (a random number determines the location of the crossover)..
- introduce some random mutations in the population. (That is change one bit in the solution)
- gradually the solutions will improve and stop when one solution is correct (as in mastermind) or when a solution is close enough (as in the min/max problem).

(Dewdney 1993a, p103)

56. Word processor

a. Line processor

Simple word processor with no wrap around or scrolling (ie: limited to one page or less)

type over, insert modes
arrow keys, home and end
insert / delete line
delete / backspace
search / replace
block move / copy
saving / retrieving
spell checker

b. More complex processor

Add to the above word-wrap, both forward and reverse, as well as scrolling. (Hard and soft returns must be differentiated.)

Math Applications

57. Letters for digits puzzles

Given 3 strings, replace each letter with one of the digits 0 to 9 to make the addition "work", as in:

```
 cat        806
+dog       +257
----       ----
rats       1063
```

58. Convert int and real to/from strings

For example: eg 23 - "23" or 3.14e-1 - "3.14e-1"

59. Convert numbers into words

Eg: 932 to "nine hundred and twenty-three"

60. Print a given integer with commas

Print 1234567 as "1,234,567", etc.

61. Roman numerals

Convert a number into its roman numeral equivalent (or reverse).

M = 1000
D = 500
C = 100
L = 50
X = 10
V = 5
I = 1

Do arithmetic with roman numerals : eg CDXLIV + DLV = CMCIX

62. Number base conversion
Convert numbers of any base to any other base (need to convert to decimal as an intermediate step)

63. Fraction Base conversion
Convert decimal fractions (numbers between 0 and 1) into ternary or other base fractions. Example:
.46 (decimal) = .110102...(ternary: base 3)
= 1/3+1/9+0/27+1/81+0/243+2/729+...
Conversion method: from decimal:

```
x = decimal fraction
while x not= 0 or max size of answer not reached
        x = x * base
        add the whole part of x to answer
        x = fractional part of x
```

to decimal:

```
power = base
for length of fraction
        answer = answer + fraction(i) / power
        power = power * base
```

64. Divisibility by 11 test
if # = 11 it is divisible by 11 else see if the number formed by subtracting the units digits from the number without the units digit is divisible by 11 (recursively). Eg

```
123456789005
12345678895
1234567884
123456784
etc.
```

Can handle VERY long numbers as strings.

65. Simple calculator
Simulate a 4-function calculator (without operator precedence) given a string such as "83+7*37=" (answer is 3330). The "=" will always be the last character.

Games

66. Create crosswords
a. given two words, print one (or all) possible cross word arrangements of those two words.
b. given a list of words and puzzle size, make a complete crossword puzzle.

67. Wheel of Fortune
Given a hidden word/phrase, spin a wheel to determine what you'll win if you can guess a consonant in the word/phrase. Vowels must be bought out of you winnings. A spin may result in landing on bankrupt. The player who solves the puzzle first wins the money he/she has collected to that point. (The wheel changes from game to game: there are 24 positions with differing amounts form 100 to 10,000 and other things such as double play, loose a turn, surprise, trips and bankrupt.)

68. Hangman
Include graphics and the ability for the computer to pick the word or guess the word. To make it fair for the computer restrict the game to a specific topic (eg: cities in North America or current baseball players, etc). If the computer is guessing, one strategy is to make several set guesses, then based on the results, search the list of words for those possible. Then guess the most frequent letter from that set and use the results to "compress" the possible list.

69. Cryptoquip
Given a phrase with all letters substituted for other letters, help the user solve the puzzle by providing a frequency list of the letters and allow the user to select a letter and its replacement then display the phrase with that substitution.
(Stephenson 1994, Gr 11, Proj)

L 2D Arrays

1. General manipulations
   a. Delete a given row or column
   b. Switch two given rows or columns

2. Min/Max Row/Column
   Given a matrix return the vector of min or max values in each row or column.

Math Applications

3. Matrix manipulation
   +, - , *
   transpose
   create identity matrix
   determinants
   Cramer's rule to solve simultaneous equations

4. Determinants
   For any column j (or row i, letting j vary)
   $\det(A) = \text{SUM}((-1)^{i+j} * a[i,j] * \det(\text{minor}(a[i,j])))$ where i=1 to n
   The minor a[i,j] = submatrix found from a by dropping the ith row and jth column.

5. Elementary row operations
   a. solve a given system of linear equations
      i. multiply a row by a value
      ii. multiply a row by a value and add result to another row
   b. find the inverse of a given matrix

6. Symmetric and orthogonal matrices
   a. determine if a given matrix is symmetric:
      a matrix is symmetric if it equals its transpose, ie: $A = A^T$
   b. determine if a given matrix is orthogonal:
      a matrix is orthogonal if its transpose is its inverse, ie: $A*A^T=I$

7. Triangular matrices
   a. determine if a matrix is upper-triangular (a(i,j) = 0 if i<j)
   b. determine if a matrix is lower triangular (a(i,j) = 0 if i>j)
   c. determine if a matrix is tridiagonal (a(i,j) = 0 if |i-j|>1)

8. Tridiagonal matrix compression
   Given a 1D array A which stores a tridiagonal matrix B and a location in B (row and column), return the value. The matrix B is stored in A using the following system: B(1,1) = A(1), B(1,2)=A(2), B(2,1)=A(3), B(2,2)=A(4), B(2,3)=A(5), ...

9. Cross multiply vectors

10. Regular stochastic matrices
    An RSM is one in which all entries are positive and the sum of each row = 1.
    a. determine if a given matrix is RSM.

b. given an RSM A, calculate A, A^2, A^3, A^4, ... until it approaches a matrix T all of whose rows are identical.
(Bent 1986, p360)

11. Gaussian elimination for solving systems of equations
Transform the system into a system with all zeros below the main diagonal, using elementary row reductions (called the forward-elimination phase). Then the values of the variables are computed using the triangulated matrix produced in the first phase (this is called the backward-substitution phase).
(Sedgewick 1988, p535)

12. Pascal's triangle
Generate each line in the triangle from the previous

13. Bell numbers
Start with a 1 in row 1. For all the other rows, bring down the last number in the previous row to the beginning of the current row. Then each succeeding number in the row is obtained by adding the previous number and the number directly above it. The last number in each row is a Bell number.

```
1
1 2
2 3 5
5 7 10 15
...
```

The Bell numbers are: 1, 2, 5, 15, ...

14. ASCL's triangle
Create the triangle of numbers: (3#'s above create number)

```
                1
            2       3
        2       6       3
    2       10      12      3
2   14      28      18      3
...
```

(For example the 18 is 3+12+3)

General Applications

15. Concentric matrices
a. given a size of a square matrix, create the following pattern:

```
eg n= 3 1 1 1
                1 2 1
                1 1 1

eg n = 5        1 1 1 1 1
                1 2 2 2 1
                1 2 3 2 1
                1 2 2 2 1
                1 1 1 1 1
```

b. Penny pitch
Use a square created above and simulate someone throwing pennies into the grid by picking random squares and adding up the points.

16. Number spirals

Given the odd size of a square matrix, fill it with the numbers from 1 to n^2 in the following order:

eg: n = 5

```
21 20 19 18 17
22  7  6  5 16
23  8  1  4 15
24  9  2  3 14
25 10 11 12 13
```

17. Magic squares (sum along each row, column and diagonal are the same)
    a. determine if a given matrix is a magic square
    b. create a magic square of given size

    (Collens 1976, p151, Brown 1983, p117)

18. Alpha magic squares

    Magic squares which are also magic if you count the number of letters in the words (of the numbers)

    eg:

    normal magic square

```
 5 22 18
28 15  2
12  8 25
```

    count of letters (also magic)

```
4  9  8
11 7  3
6  5 10
```

    both squares are magic!

    Also includes the idea of "log(x)", meaning the number of letters in the words for the number.

    (Scientific American, Jan 1997, p106)

19. Latin squares
    a. determine if a given square matrix of size n is a latin square (each row and column is a permutation of a set of n distinct symbols)
    b. determine if 2 given latin squares of the same size are orthogonal (when the two are superimposed, each element of the first square occurs once with each element of the second. That is, each pair only occurs once.)
       (Dyck 1984, p301)
    c. create all possible latin squares in "standard form" of a given size.
       (Collens 1976, p151)

20. Double Letter Combination Count

    Create a double letter frequency count (arranged as a 2D table), called a "digraph" (ie: count the number of times "aa", "ab", "ac", ... "zz" occurs). Include "_a" or "a_" ("_" is a blank)

21. Mountain passage

    Given a grid of elevations, start at top left corner and go to bottom right corner. Can go in any of 4 directions but not if the difference in elevation is > 2. Any number greater than the starting number requires 1 unit of oxygen per step (use 1 unit if start or stop step is > starting step). Print whether a route is possible and the number of oxygen units consumed. Example:

5 4 3 2 1
7 **5 6** 6 6
8 **8 8** 9 6
9 **6** 9 9 6
4 **5 4 5 3**

    This route takes 5 units of oxygen.

22. Transferable vote elections

This is a system where a successful candidate requires the support of a majority even if there are more then 2 candidates. Each voter votes for all candidates, in order of preference. (These votes are input into 2D array.) The first choices are counted in round 1 of voting. If there is no winner, the candidate with the fewest votes is eliminated. On the second round of voting the second choice is taken for those voters who voted for the eliminated candidate. This process continues until there is a winner or there is a precise tie.
(Findlay 1985, p287)

23. Sharks and fish (Wat-Tor world)
This is a simulation of ecological rhythms in a simple predator/prey relationship. Inputs include breeding times, starvation times, initial population sizes. The position of the sharks and fishes are shown after each time period, along with other statistics.
(Dewdney 1988, p239)

24. Neural networks
    a. design a simple neural network to convert polar to rectangular coordinates
    (Dewdney 1993b, p46)
    b. design a pop machine simulation.
    (Algorithm 2.1, p8)

25. Simple spreadsheet
Create a simple spread sheet program.
(Brown 1988, p148)

26. Average prices
Let the columns be the months and the row be the years and record the price of an item in the table, then determine the average month or year prices.
(Hume 1990, p76)

27. Class marks program
Store and manipulate a set of marks for a class: need arrays for student names and information, mark descriptions, weightings and categories and a 2D array for the marks themselves. The program should allow input and changing of marks and other information. It should compute averages and final marks.

28. Distance between cities
    a. set up a matrix of distances between cites, then given a series of cities, determine the total distance
    b. find the shortest distance between two cities:
    Assume there are n cities. For all cities that are not directly connected, store a VERY large number. Then let k vary from 1 to n. For each iteration of k, go through all combinations of cities i and j and set the new distance between i and j = minimum of (old distance between i and j) and (old distance between i and k plus the old distance between k and j).
(Solow 1988, p494)

Fractals (and Cellular Automata)

29. Cluster size determination
Given a 2D array A of 0's and 1's (empty and full) and an initial square (i,j), return the size of the cluster containing the square:

```
if A(i,j) = 0
        return 0
```

else

set A(i,j) = 0

return 1 + Count in the surrounding 4 or 8 squares (depending on the definition of "neighbourhood").

30. Clusters

Given the grid size and the percentage of filled squares, identify the distinct clusters within the grid (and colour them differently) and count the number of clusters of each size. Plot the resulting counts on a log/log scale.
(Kaye 1989, p206)

31. Fractal forest fires (percolation in clusters)

Create a grid filled with x% of live trees. Set all trees on one edge on fire. After each hour (or minute or pass) set the neighbouring trees on fire (either in all 8 directions, or orthogonally). The forest "percolates" if the fire travels to the opposite edge. An alternative approach is to light the fire at some random tree in the grid.
(Pietgen 1991a, p386)

32. Crystal growth by Diffusion Limited Aggregation (DLA)

A seed particle is placed in the centre of the screen (or a circle of seed particles can be placed around the outside, or a line of them can be placed along on edge of the screen). Then particles are launched from far away and allowed to travel randomly. If it arrives at an adjacent site to a seed particle (or any particle in the crystal) it will "stick' and become part of the crystal.
(Lewis 1990, Chap5, Kaye 1989, p245, Schroeder 1991, p194, Peitgen 1992a, p399, Dewdney 1988, p116)

33. Sandpiles: self-organized criticality

Both 1D and 2D simulations are possible. Either the pile is built up one grain of sand at a time, or a wet pile is allowed to dry and "relax". The idea is that in all cases the height difference between 2 points cannot be greater than some fixed critical value. If they are the sand will tumble down.
(Bak 1988)

34. Sandbox simulation

Create a computerized sandbox. Graphically show what happens as 1 grain of sand is added to the centre of the pile. One 2D array will keep track is the height of the pile at that point (use different colors for different heights), and another shows movement (slides).
(Algorithm 1.5, p7)

35. Simulate nuclear chain reactions

Given a grid size (or size of atom), draw the atoms (of "gridium") on a 2D grid and "explode" the central atom. This will send out two neutrons which may cause other atoms to explode. Find the grid size necessary to get x% exploded.
(Algorithm 4.2, p12)

36. Life

This is an example of a 2D cellular automata. On the grid, each cell can either be alive or dead. Every time period, each cell may or may not change its state depending on its eight neighbours and the following rules:

1. if a cell is alive at time t, it will remain alive at time t+1 if it is not overcrowded or undernourished, ie. if there >3 living neighbours, or <2 living neighbours.
2. if the cell is dead at time t, it will remain dead unless it has exactly 3 parents (ie: 3 living neighbours)

The initial population of cells can be stored on file, randomly generated or input by the user (type in the coordinates or graphically select).
(Dewdney 1989, p271)

37. Other cellular automata
Variations on the game of Life, with different rules are possible.
(Pietgen 1991b, p72)

38. Langton loops
Self-reproducing loops (there are several types)
(Algorithm 3.3, p16)

39. Tur-Mites
    a. a 2D version of the Turing machine generates some interesting patterns on a 2D grid.
    (Dewdney 1993b, p59)
    b. try two turmites on the same grid.
    (Algorithm 4.1, p21)

40. Busy Beavers
Simulate a Turing machine.
(Dewdney 1988, p 160)

Games

41. Word search
    a. given a series of words and a puzzle size, randomly place the words in the grid (in any of the 8 possible directions). Fill any remaining spaces with random letters.
    b. given a 2D array of letters, the computer should find given words in the array. The words can be horizontal, vertical, or diagonal (and in either direction).
    c. allow the user to find words in a computer created word grid, by allowing the user to type in the coordinates of the first and last letters of the word and then highlighting the word in a different color.

42. Triangle peg game
Given a triangular pyramid of pegs, 1 on top, 2 on next row, etc (5 rows). All but top has pegs. Pegs can jump over one other and remove the jumped peg. Winning position is to get a single peg left.

43. Mazes
Find the path from the start to the finish:
A path exists from a given spot if that spot is the finish or if there exists a path from any of the adjacent spots. A path does not exist if the spot is a wall or a stop or "already visited". (Thus this requires a grid with each spot marked with either empty, start, finish, wall or "already visited". If a path is found it will be the "already visited" spots. Note this will NOT likely be the shortest path. For the shortest path always force the recursion to check the spot which is closest in direction to the finish first.)
(CCC 1998)

44. 15 puzzle
Fill a 4x4 array randomly with the numbers 1 to 15. There will be one empty space. Allow the user to move adjacent numbers into the space and count the number of moves it takes for the user to solve the puzzle and return the numbers to ascending order.

45. Electronic chess/checker board
Draw the board and place the pieces on the board. Allow users to move the pieces. Program should maintain scores (total number/value of pieces on the board) and detect winning situation (for checkers at least).

46. Chess problems
Given the locations of the chess pieces on the board determine:
a. if any piece is attacking a given piece
b. the number of squares (or the squares themselves) that a given piece can move to

47. Tic-Tac-Toe
Given the board and positions of the X's and O's, determine if anyone won. Play the complete game.

48. Hexapawn
A combination of chess and tic-tac-toe!
(Brainerd 1982, p536)

49. 4 in a row
Like tic-tac-toe, but there is a grid of 6x7 openings and you drop disks (one color for each player) from the top into one of the 7 columns. First t get "4 in a row" wins.

50. Othello
Played on an 8x8 board with four pieces arranged as:

WB (in the center)
BW

Each move consists of placing a piece such that you "bracket" (surround) the opposite color, and then they become your color. The winner is the one with the most pieces of his color left.
a. given a position, determine the best move
b. play the complete game (computer one player)

51. Sim (game)
Given 6 dots on the screen, players alternate drawing lines between them (in two different colors). The first player to complete a triangle loses. Use a 6x6 array to store moves (eg: if a player connects dot 3 and 6, then set A(3,6) = A(6,3) = 1.
(Bent 1986, p319)

52. Dots and Boxes (making squares game)
Given a grid size, players alternate drawing single lines between the dots. The person who completes a square "owns" the square (that player can then play again). The game ends when there are no more lines to draw and the winner is the player who 'owns" the most squares.

53. SOS
Like dots and boxes in a way. There is a big grid and player alternate writing S or O in a box. Either player can draw either an S or O. 1pt for each SOS created. Continue until all squares filled. Winner has most points.

54. Bingo
A bingo card has columns B, I, N, G, O and random numbers in the ranges (1-15, 16-30, 31-45, 46-60, 61-75) down each column, with a "free" in the centre.
a. create the cards
b. evaluate a card given the numbers called
c. play a complete game

55. Boggle
Given a square of letters find words, moving from one letter to another in any of four (or eight) directions. Can't repeat a letter. Can use various "rules" as to what makes a word, or you can use a list of words, etc. Example the words "age" and "roam" and others can be found in this square:

```
a s n d
g e r o
a f l a
p s c m
```

56. Black Jack
If the computer uses a table of probabilities and recommended strategies, then a 2D array is required for this version of the game.
(Tamburin 1993, p5)

57. Light bulb game
solitaire game. Given a 5x5 grid of lights (randomly turned on/off) each controlled by a button which toggles the respective light and the surrounding non-diagonal lights. Objective to turn off ALL lights.

58. Knight's tour
Can the knight chess piece move around an empty chessboard and touch each of the 64 squares once and only once?
(Deitel 1994, p260)

M Multi-D Arrays

1. Price matrix
   Let the columns represent the months, the rows represent the years and the 3rd dimension represent the grade of the product, record the price in the table. Then use this to determine various things. (Hume 1990, p78)

N Sorting

See also: Searching ideas (because the most common type of sorting question is a database type question which may involve both sorting and searching.)

1. Basic slow methods ($O(n^2)$)
   a. bubble
   b. insertion
   c. selection

2. Basic fast methods (O(nlogn))
   a. merge
   b. quick (Ammeraal)
   c. shell
   d. comb (Arnush 1995, p183)
   e. heap
   f. radix or bucket (Deitel 1994, p264)

3. Basic sort modification
   Almost all sorts can be modified to
   a. reverse the sort order, ie. ascending/descending
   b. simply make the algorithm work starting from the reverse end of the array
   c. sort multiple arrays or structures together

4. Recursive insertion sort
   Algorithm: (array a from 0 to n-1)

```
for i: 1 to n-1
        Insert (i, a(i))

Insert (i, v)
    if i = 0
        a(0) = v
    else if a(i-1) > v
        a(i) = a(i-1)
        Insert (i-1, v)
    else
        a(i) = v
```

   (U of Waterloo CS134, Fall 99)

5. Recursive selection sort
   Given an array x and logical size n

```
if n > 1
        exchange largest in x with x_n
        Selection_Sort (x, n-1)
```

6. Quick sort variant

```
sort (left, right)
    swap (left, (left+right)/2)
    lastsmall = left
    for i: left+1 to right
        if a(i) < a(left)
```

```
                    lastsmall++
                    swap (lastsmall, i)
        swap (left, lastsmall)
        pivotplace = lastsmall
        if left < pivotplace - 1
            sort (left, pivotplace - 1)
        If pivotplace + 1 < right
            sort (pivotplace+1, right)
```

(Hume 2000, p 498)

7. Heap sort

```
Insertheap (i, x)
    child = i
    a(i) = x
    while (true)
        parent = child / 2
        if child = 1 or a(parent) > a(child)
            break
        swap (parent, child)
        child = parent

Removelargest (n)
    largest = a(1)
    a(1) = a(n)
    parent = 1
    while (true)
        child = 2* parent
        if child > n break
        if child < n && a(child+1) > a (child)
            child ++
        if a(parent) > a(child)
            break
        swap (parent, child)
        parent = child
    return largest

HeapSort
    for i: 2 to n
        Insertheap (1, a(i))
    for i: n to 2
        Removelargest (i)
```

(Hume 1998, p365)

8. Quadratic selection sort:
   - divide array into 4 groups of 4, each n/4 elements
   - find smallest in each group and place in a temp array of 4 elements
   - take smallest of these four, and replace that one by the next smallest from the original subgroup.

   How does this compare in terms of speed and space with original selection?
   (Hume 1990, p229)

9. Sorting by counting

a. given an array x of size n, create an index array I. Set a count to 1 and go through x, counting the number of times the first item is >= the other items. Then set I(count) = 1. Reset the count to 1 and count the number of times the second item is >= the rest of the items. then set I(count) = 2. Repeat for all the items. (In general for the k^{th} item, set I(count) = k.) Then the sorted list is obtained by using x(I(k)), k = 1 to n.
(Dwyer 1984, p211)

b. declare an array count, set count(i) = number of elements <= x(i), then place x(i) in position count(i) of an output array. Watch out for duplicates!
(Tenenbaum1981, p385)

10. Distribution counting sort
Similar to above idea.
(Sedgewick 1988, p111, Tenenbaum1981,p386)

11. Factorial sorting
Randomly scramble the elements in an array (for each element i, let j be a random value 1 to n, exchange elements i and j.) Repeat until the array is sorted (by chance so to speak) Determine the number of tries required. Is it > or < the factorial of n?

General Applications

12. Sort comparisons
a. compare times of various sorts
b. find the minimum file size such an O(n log n) sort becomes more efficient (time wise) than a $O(n^2)$ sort.

13. Sub sorting
Given an array of structures, perform a sub sort. For example, sort first names within equal last name groups.

14. Graphically represent sorts
Draw lines to represent the numbers being sorted and as they are moved by the sort, move them on the screen.

15. Bubble sort improvements
Make sure it exits when array is sorted (ie: no more exchanges) and sort in opposite directions on each pass (Shaker sort)

16. Combine sorts
For example: use quick sort to sort a large array, but when the pieces get to be less than x long, use insertion sort.
(Sedgewick 1988, 125)

17. New sorting order
Given a new sorting order eg : 4,6,8,9,7,2,3,1,0,5 sort a list of numbers. For example the following is sorted according to the above order: 44685, 845, 896, 1234, 1345

18. Sort Numbers alphabetically
Given a series of numbers from a to b, sort those numbers by string value. Eg the numbers 7 to 10, sorted alphabetically are 8,9,7,10.

19. Tree sorting

Create a binary search tree, then do an in-order traversal and recreate the array. Compare speed with other sorts.

20. Basic statistical functions
After sorting, find:
    a. mean
    b. median
    c. mode

O Searching

1. Basic methods
   a. linear searching
      i. normal method:

```
i = 1
while  i <= n and x[i] not= key
        i = i +1
```

      ii. better method using a sentinel

```
i = 1
x[n+1] = key
while  x[i] not= key
        i = i +1
```

   b. binary searching
   c. ternary searching (same as binary, only split the list into 3 at each stage)

(Bentley 1986, p85-90)

2. Recursive linear search
   Given an array x, logical size n and key

```
if n = 1
        return x1 == key
else
        return xn == key or Search (x, n-1, key)
```

General Applications

3. Search comparisons
   a. for a particular file size and number of random searches, compare the times for various searches.
   b. find the minimum file size when the binary search becomes more efficient than a finely tuned linear search.

4. Graphically show the progress of the search
   For each different search method, graphically show the progress of the search.

5. Dictionary
   Given a line(s) separate into words and search a dictionary (a file of words) for them. Allow: skip, add, replace.

6. Dictionary extended (spell checker)
   On a misspelled word, find the word(s) that most likely are ones the user was trying to spell: eg: find words with the same letters, but with only a couple rearranged (“waht” for “what”), or find words with one extra or missing letter(“wat” or “whhat” for “what”), or words with only one or two incorrect letters (“wgat” for “what”), or with double letters (“whhat”for “what”).

7. Find words in a list, one letter at a time
   Display a big list of sorted words or phrases, starting with the first word. As the user enters letters, jump to the section of the list corresponding to the partial word typed. The list should always start with the first word that satisfies the partial word, or if there is no match, the list should be positioned at the word before where the user’s word would go if it were in the list. Backspace should be

handled as well. (This should word like WordPerfect's find word in the dictionary or Encarta's topic list.)

8. Union/Intersection of sets
Given to unsorted arrays A and B, find:
- all elements in both A and B (no duplicates) (intersection)
- all elements in sets A and B (no duplicates) (union)
- all elements in A, not in B

9. Anagrams
   a. given a LARGE dictionary (one file of many tens of thousands of words), solve single and multi word anagrams.
   (Dewdney 1988, p 200)
   b. find all sets of anagrams (eg "pots", "stop" and "tops")
   The method is to sort the letters of each word, and then store both the sorted letter word and the real word itself. Sort the file of these structures based on the sorted letter words and the anagrams will show up as duplicates in this file.

10. Self-organizing sequential search
Move the last item found to the front of the list and move all others back one. Is it more efficient than a straight sequential search in some or all situations?

11. Indexed database
Store records in a main database (a random access file) in random order, or just the order that they are added. Maintain an index table (loaded into an array in memory) that holds the keys for each record, and their locations. This will be sorted by key. To find a record, simply look up the key in the table using a binary search and then use the location to access the main file. Use this idea to maintain any sort of general database. (Note: Deletion is handled simply by deleting the index entry or by maintaining a tagging system and the main database needs to be updated only once in a long while.)

Data Base Applications

12. General database manipulation
Any kind of database imaginable makes a good structure exercise. Make it relevant/interesting to the students (or let them choose the content of it) and the complete program should be able to:
   add record
   delete record
   change record
   display many records at once and allow scrolling
   find a record
   sort the records
   etc.
Examples of such databases include:
   address / phone book
   chemistry database
   baseball statistics
   astronomy data
   appointment calendar

13. Locker Database
Include locker number, student name(s), lock serial number, lock combination.

14. Event database
Problems involving an event database can be created using an event structure which is made of both date and time structures.

15. Sports databases
a. team scheduling
b. team tracking: statistics
c. athletic letters

16. Birthday database
Database program using an array of date structures.

17. Stock market database
Include company name, closing stock price today and yesterday, number traded. The program could find the top 10 decliners, gainers, most active, etc.

18. Store inventory database
Maintain information on the stock number, description, cost, list price, quantity, etc. Allow processing options of addition, deletion, change, sales, re-order/re-stock, stock taking reports, etc.

19. Hardware maintenance and repair database
Track inventory (include all relevant details for each piece of equipment including current location) and include past and present repairs and problems.

20. Royalty database
A store which sells products which generate royalties for the inventor/artist (eg: games or CD's) could maintain a database of information about the items as well as the royalty holder. The program could maintain the database as in a regular store database, plus produce reports showing royalties due, etc.
(Solow 1988, p538)

21. Customer record database
Each record should contain: name, address, account #, account status (current, overdue, delinquent), old balance, current payment, new balance, payment due. Allow batch mode and/or interactive processing for various transactions, such as:
adding/deleting customers
new bills
payments
reports on delinquent/overdue accounts
billing slips
displaying customer information
NOTE: arrays need not be used, this could be a pure file handling problem.
(If the current payment > 0 and <10% of old balance then the account is overdue, if the current payment = 0 then it is delinquent, otherwise it is current.)

22. Airline reservation system
Need two databases: one for the flights (flight number, depart city, arrive city, departure time, arrival time, number of 1st class seats, number booked, number of regular seats, number booked, etc.) and one for the reservations (name, flight number, class, cost, etc.). Must be able to add/change/delete flights as well as reservations. Various reports need to be produced such as passenger lists.

23. Library circulation database

Store book titles, catalogue number, cost, publisher, etc. Also store patron information: name, address, books on loan, due dates, fines, etc.

24. Lost and found database system
Maintain two databases of lost and found items, including item type, descriptions, owners, finders, value, date lost/found, etc. Try to match lost and found items automatically.
(Carter 1989, p538)

P Hashing

1. Basic methods
   Implement for some database. Show graphically where the keys go.
   a. linear-probing
   b. double hashing
   c. chaining

2. Comparing load factors and sizes
   Using linear probing and the hash function of H(k) = k mod M (where M is the number of entries in the table), use different load factors (# of entries/M) of 0.5, 0.7 and 0.9 and different M's (100, 200, 300, etc.) and then for each combination, after loading the table, generate 500 random # to find in the table and record the average # of comparisons needed.
   (Carter 1989, p367)

Q Linked Lists

1. General linked list manipulation
   Maintain linear or circular (with or without header node), singly or doubly linked lists (using dynamic storage and pointers). Basic operations: insert/delete node, find, empty, traverse, etc.

2. Exchange nodes
   a. given pointers to 2 nodes in a list, exchange them.
   b. exchange the m^{th} and n^{th} nodes in a list

3. Delete last node in the list

4. Delete the nth node in a list

5. Delete every second node in a list

6. Delete all the nodes in a linked list

7. Reverse all the nodes in a linked list

8. Join (concatenate) two linked lists

9. Make a copy of a linked list
   This can be done with a recursive node clone method (Java) or can be done with the following recursive algorithm:

```
Copy (node a)
        if a = null
                return null
        else
                p = new node (a->data)
                p->next = copy (a->next)
                return p
```

10. Move the pth node, n positions forward in the list

11. Print the last n nodes in a singly linked list
    The trick is to maintain two node pointers, n nodes apart

12. Split circular linked lists
    Given a circular linked list (with or without header node) with an even number of nodes split the list into 2 circular linked list each with half the number of nodes. Can do it by counting and one list is fist half and the other the second half, or you can make 2 lists of alternating nodes.

13. Loops in a linked list
    determine if a linked list has a loop (that is, does the last node in a singly linked list point to an intermediate node?)
    Method 1: use two pointers, fast and slow. Fast moves through list twice as fast as slow. If fast equals null then no loop is fast ever equals slow then loop.
    Method 2: use a flag bit. Traverse the list and set to true. If you find a flag set to true you have a loop.

14. Self-organizing linked list
During a search, after the node is found, place it at the front of the linked list

General Applications

15. Josephus problem
Given N people standing in a circle, kill the mth person going around the circle. Close ranks as each person drops form the circle. Find out who is the last person to remain alive and also find the order of the people killed.
(Sedgewick 1988, p21)

16. Polynomial manipulation
Represent polynomials with a linked list and create a complete manipulation program (input, output, evaluate, add, subtract, multiply, find derivatives, integrals, roots, plot, etc.)

17. Big numbers
Represent big integers with a linked list and create a complete manipulation program (input, output, add, subtract, multiply, etc.)

18. Ropes
Ropes are long strings, implemented with either arrays or linked lists. Perform basic operations on ropes such as: printing, conversion between strings and ropes, concatenate ropes, extract subropes, etc.
(U of T 95)

19. Music editor
Allow the user to enter music scores, play them, save them on disk or to edit them. A linked list is used to manipulate the music.
(U of T, 1995)

20. Dual pointer mailing list database
Create a mailing list able to display entries in order either by name or postal code. Information includes name, street, city, province, and postal code. Need two sets of links to connect to next name and to next postal code.
(Koffman 1994, p329)

21. Sparse matrices
Basic operations: insert element, print matrix, add, multiply, transpose, etc.

22. Freeway simulation
With doubly linked lists, maintain a simulation of a multilane freeway, with many cars. Graphics could be included.
(Algorithm 3.3, p12)

R Stacks

1. Brackets
   Check expressions for correct bracket placement. (Use {}, [], () brackets)

2. Check string for form xCy
   Determine if an input string is of the form xCy, where x is any string of "A" and "B"'s and y is the reverse of x. (Only allowed to process one character at a time!)

3. Infix / postfix conversions
   Infix to postfix conversions using expressions of single digit operands and binary operators

4. Postfix evaluation (full expression evaluation)
   a. evaluate a postfix expressions using only single digit operands and binary operators.
   b. evaluate a full infix expressions with multi digit operands and unary operators (including trigonometric functions, etc), by combining the infix to postfix conversion and the postfix evaluation.

5. Reverse a list

6. 2 Stacks in 1 array
   Design a way to keep 2 stacks in one array such that neither stack overflows until all elements of the array are used and neither stack gets shifted. Hint: Have the stacks grow from opposite ends of the array.

7. Simulate Recursion
   Use a stack to simulate the recursive formula:

   f(n) = 1 if n <=1
   = 2n + f(sqrt(n)) + f(n/3) if 1< n <= 100
   = 3n + f(n/20) + f(n-10) if n > 100

   (Queens U, assn#2, CISC124, 2001fall)

S Queues

1. Basic manipulation of queues:
   insert, delete, empty, full
   with various implementations:
   Arrays
   straight line
   shifted back on each remove
   shift back on full
   circular
   Linked lists
   Use graphics to show what is happening.

2. Basic manipulation of a deque (double-ended queue)
   Elements are inserted and removed from both ends
   insert right/left, delete right/left
   With a linear implementation and arrays explore shifting options and full right/left

General Applications

3. Radix sorting
   Need 10 queues, 0 thru 9, plus an eleventh queue for gathering. There is one pass for every digit in the numbers to be sorted. During the first pass, look at the units digit and each number is added to the rear of the appropriate queue (ie: 45 would be added to the 5 queue). After each pass, the numbers from each queue, beginning with 0, are copied to the rear of the eleventh queue. Repeat for next digit.
   (Koffman 1994, p329)

Simulations

4. Simulations of queues in a store/bank
   For each customers store the time waited and the time to process the order. For each server store the number in line, the total free time and the time to process the current order. The servers are simply an array, but the customers should be stored in either a single or multiple queues. Allow the user to see the various quantities as they change, graphically display the size of the queues, etc. and allow the user to modify such things as the probability of arrival or maximum order time.

5. Cashiers and express lanes
   Multiple cashiers each with their own queue. Customers have x items (1 to 2N). Time to process is 6x+20 seconds. Inter arrival time is 50 sec. Mean # of items is 20 (ie. N = 20). Run simulation with or without express lane (cashier for customers with < 10 items). Calculate avg wait times, avg queue length and cashier utilization in both simulations. Queues can be separate objects or part of cashier object.
   (U of Waterloo CS134, Assn#5, Jul 2000)

6. Simulation of kids at houses on Halloween
   (U of Waterloo CS134, Fall 2001)

7. Airline ticket agent
   With one agent, there are 2 queues, one for regular customers and another for first-class. The first-class customers are serviced more often than regular. Simulate this system and track waiting times

for the various customers and experiment with various number of first class customers that are serviced before switching to the regular queue.
(Koffman 1994, p246)

8. Airport simulation
Simulate landings and takeoffs, with one or more runways. (Aircraft are queued up on the ground and in the air: those in the air get preference.)
(Koffman 1994, p260)

9. Printer queues
OS assigns print jobs to different queues depending on number of pages to be printed (0-10, 11-20, 21-50, etc). Smaller jobs are printed before larger. Assume one printer can print 5 pages per minute. Print requests come every x minutes with a variable number of pages to print. There will be 100 requests in total. Output order of jobs received and printed, waiting times and total time to print all 100 jobs. Compare with 1, 2, or 3 printers on the system.
(Koffman 1994, p260)

Games

10. Snake (game)
A queue holds the coordinates of the snake, (head is at the rear). Player moves the snake around the screen, eating food (a number indicating how long the snake will grow). The game is over when the snake runs into itself. Object is to get as long a snake as possible. Moving the snake is easy, move the head in the correct direction (ie: add to the queue) and erase one unit from the tail (ie: remove from the queue). In the case the snake has just eaten a number, wait that many moves before erasing.
(U of T, 1995)

T Trees

1. Basic manipulation of binary search tree
   a. insert, find, empty
   b. delete (this is difficult)

2. Tree traversals
   Traverse tree in pre, in and post orders.

3. Print the tree breadth-first order
   Print the nodes of a tree in breadth-first order, that is all nodes on one level at a time, left to right. The method is to use a queue:

```
put the root on a queue
recursively:
    if the queue is not empty
        take from the queue and print
        add the left child to the queue
        add the right child to the queue
        call the recursive print routine
```

4. Count the nodes in a tree

5. Count the leaves of a tree

6. Find height of a tree
   a. height of a leaf is 0, height of a tree is one more than the maximum height of the left and right subtrees. OR
   b. height of an empty tree is 0, height of a tree is one more than the maximum height of the left and right subtrees.

7. Find level of a node
   The level of the root is 0, the level of any other node is 1 more than the parent.

8. Copy a tree
   Write a method which will create a new tree which is a copy of the current tree.
   (See Mirror copy algorithm for how to do it!)

9. Mirror copy a tree
   Write a method which will create a new tree which is a mirror copy of the current tree.

```
Node Mtree (node t)
    if t = null
        return null
    else
        temp = new node (t->data, Mtree(t->right), Mtree(t->left))
        return temp
```

10. Tree Deletions
    Given a binary search tree, find and remove the min / or max value in the tree (this is the first stage of a full delete.)

11. Path lengths

a. find path length of the tree
= sum of the levels of all nodes in the tree
b. find internal/external path lengths of the tree
Internal nodes are non-leaf nodes, external nodes are leaves.
(Sedgewick 1988, p37)

12. Similar trees
a. determine if 2 trees are similar (they are if both are empty or both the left and right subtrees of each are similar)
b. determine if 2 trees are mirror similar (they are if they are both empty or if the left subtree of one tree is similar to the right subtree of the other tree and reverse.)

13. Siblings
Given a pointer to a tree and a node in the tree, return a pointer to the node's sibling.

14. Youngest common ancestor
Given a pointer to a tree and 2 pointers to non-root nodes of the tree, return a pointer to the youngest common ancestor of these two non-root nodes.

15. Determine if a tree is height-balanced (an AVL tree)
A tree is height balanced if for all nodes in the tree, the height of the left and right subtrees differ by at most 1.
(Ammeraal 1992, p 138)

16. Determine if a tree is complete
A tree is complete if all leaves in a tree with n levels (0 through n-1) are at level n-1 or n-2, with all leaves in level n-1 as far left as possible.

17. Heaps
A heap is a complete tree in which for all nodes, the key is larger than that in its children.
determine if a tree is a heap
insert and delete from the heap
(Koffman 1994, p382)

18. N-ary trees
Create a binary tree which represents a n-ary tree and explore the effect on the traversal order (pre-order traversals are the same, but in-order on binary tree is the same as post-order on n-ary tree.)

General Applications

19. Concordance
a. read in some text, store individual words and produce a cross-reference or index, listing all words and all line numbers where the words occurred (one way to do this would be to have each node of the tree contain pointers to a queue of line numbers)
b. extend this to include major and subterms!
(Tenenbaum1981,p274)

20. Very simple language interpreter
Define expression, term and factor recursively. Read an expression in correct infix notation with brackets if necessary, and output its value.
(Ammeraal 1992, p234)

21. Mathematical expression binary trees
Store the operands in the leaves and operators as parents. In-order gives the infix expression, and post-order gives the postfix expression and therefore a method to evaluate.

22. Simple symbolic algebra simplification
Read and simplify expressions. Only the basic 4 operations are used, single digit constants and single letter variables. Simplify things such as x+0, x*1, etc. (those things involving 0 or 1).

23. String concatenation tree
Using the operators of + (concatenation), - (removes) and @ (starting at) create and then evaluate a binary tree.
"a" + "bcd" = "abcd", "abcd" - "bc" = "ad" and "abcde" @ "bc" = "bcde"
Expressions are input in postfix notation. When reading: if a string, create a tree of height 0 and push tree onto stack: if operator, pop twice and create a new tree with op as root and the two popped trees as left and right subtrees, push resulting tree onto stack. At end the only element is the tree. Can include variables as well as string constants.
(U of Waterloo, CS134, Ass #7, 12/7/2000)

24. Morse code in binary tree
Let every dot cause a branch to the left and ever dash a branch to the right. Each node would contain the letter represented by the code formed by tracing a path from the root to that node. After the tree is created, read a code (use a space between each letter and two between each word) and translate into English.

a(.-)	b(-...)	c(-.-.)	d(-..)	e(.)	f(..-.)	g(--.)
h(....)	i(..)	j(.---)	k(-.-)	l(.-..)	m(--)	n(-.)
o(---)	p(.--.)	q(--.-)	r(.-.)	s(...)	t(-)	u(..-)
v(...-)	w(.--)	x(-..-)	y(-.--)	z(--..)		
1(.----)	2(..---)	3(...--)	4(....-)	5(.....)	6(-....)	7(--...) 8(---..) 9(----.) 0(-----)

(Koffman 1994, p378, Deitel 1994, p342)

25. Water Jug Problem
Given two jugs capacity of x and y litres. Determine the steps necessary (if possible) to get z litres in one jug. (Only allowable steps are fill or empty either jug or pour x into y or vise versa.) Complex tree search using queues.
(SMSU pp#4)

26. Family trees
Create a tree with person's name and sex and have pointers to father, spouse, first born and younger sibling. Accept a pointer to a person and then output the person's children, ancestors, descendants and patriarchal descendant family tree.
(Hume 1990, p253)

27. Quad trees and image compression
Divide the image into quadrants and each of these into quadrants, etc. Each node will thus point to 4 subtrees. The process stops when the quadrant is a solid color or a pixel. Aside from simple storage and retrieval, these quad trees can be rotated, scaled or translated.
(Dewdney 1989, p290)

28. File compression
Use Huffman encoding
(Sedgewick 1988, p323)

U Misc Project Ideas

1. Pretty print
   Create a pretty print program to properly format a C (or any language) program. ie: fix the indentation and line and character spacing.

Index

References

** indicates an excellent source of ideas*

*Algorithm: The Recreational Programming Magazine. Ed. A.K. Dewdney. Vol 1.1 Nov/Dec 1989 - Vol 4.2 June 93.

*Ammeraal, Leendert. Programs and Data Structures in C. (2nd Edition) John Wiley & Sons, Toronto. 1992.

Arnush, Craig. Teach Yourself Turbo C + + 4.5 for Windows in 21 Days. Sams Publishing, Indianapolis. 1995

Bailey, Edward E., Boychuk, Schellenberg & Werezak. Computer Science: A Structured Approach. DC Heath Canada Ltd. 1985.

Bak, Per &Tang, Chao & Wisenfeld, Kurt. "self-Organized Criticality", Physical Review A, July 1, 1988.

Bent, Robert J. and Sethares, George C. Basic: An Introduction to Computer Programming (3rd Edition). Brook / Cole Publishing, Monterey, CA. 1986.

Bentley, Jon. Programming Pearls. Addison-Wesley, Don Mills. 1986.

Bentley, Jon. More Programming Pearls: Confessions of a Coder. Addison-Wesley, Don Mills. 1988.

Bielig-Schulz, G. & Schulz, Ch. 3D Graphics in Pascal. John Wiley & Sons, Toronto. 1990.

Borse, G.J. FORTRAN 77 and Numerical Methods for Engineers. PWS Engineering, Boston. 1985.

Brainerd, Walter S. & Goldberg, Charles H. & Gross, jonathan L. Pascal Programming: A Spiral Approach. Boyd & Fraser Publishing, San Francisco. 1982.

Brown, Marc H., Casci, Donald G., McGoldrick, Janice A. & Tebrow, Gerald I. American Computer Science League Book I: 1978 - 1983 Senior Division. ASCL, Providence RI. 1983.

Brown, Marc H., Casci, Donald G., Meegan, Janice A. & Tebrow, Gerald I. American Computer Science League Book E: 1983 - 1988 Senior Division. ASCL, Providence RI. 1988.

Carter, John. Problem Solving in Pascal. Addison-Wesley, Don Mills. 1989.

*Collens, P.A., Collens, R.J., Peterkin, C.R., Scuse, D.H., Stanton, R.G. Structured Fortran with WATFIV-S and WATFOR-11S. Charles Babbage Research Centre, Winnipeg. 1976.

*Deitel, H.M., Deitel, P.J.. C++ How to Program. Prentice Hall, New Jersey. 1994.

*Deitel, H.M., Deitel, P.J.. Java How to Program. Prentice Hall, New Jersey. 1999.

*Denning, Dorothy E.R. Cryptography and Data Security. Addison-Wesley, Don Mills. 1982.

*Devaney, Robert L. Chaos, Fractals & Dynamics: Computer Experiments in Mathematics. Addison-Wesley, Don Mills. 1990.

Dewdney, A.K. "Mathematical Recreations: Random Walks that Lead to Fractal Crowds", Scientific American. Dec 1988, p.116.

Dewdney, A.K. "Mathematical Recreations: Tools for Computer Graphics Make an Invisible World Seem Less Alien", Scientific American. Jan 1991, p.118-121.

*Dewdney A.K. The Armchair Universe: An Exploration of Computer Worlds. MacMillan, Toronto. 1988.

Dewdney, A.K. The Turing Omnibus: 61 Excursions in Computer Science. Computer Science Press, Rockford MD. 1989.

*Dewdney, A.K. The New Turing Omnibus: 66 Excursions in Computer Science. Computer Science Press, New York. 1993a.

Dewdney, A.K. The Tinkertoy Computer and Other Machinations. W.H. Freeman & Company, New York. 1993b.

Detar Delos F. Principles of FORTRAN Programming. W.A. Benjamin, Menlo Park CA. 1972.

*Drake, Bob. Programming Applications. Copp Clarke Pitman, Toronto. 1988.

Dwyer, Thomas & Critchfield, Margot. BASIC and the Personal Computer. Addison Wesley, Don Mills. 1984.

Dyck, V.A., Lawson, J.D., Smith, J.A. Fortran/77: An Introduction to Structured Problem Solving. Reston Publishing, Reston VA. 1984.

Findlay, William & Watt, David A. Pascal: An Introduction to Methodical Programming. Pitman, Toronto. 1985.

Frey, Richard L. According to Hoyle. Fawcett Crest, New York. 1970.

Gershtein, Sergey, Internet Forum: rec.puzzle, message dated 4 Nov 1995, Topic: "Programmers Competition"

*Giblin, Peter. Primes and Programming: An Introduction to Number Theory with Computing. Cambridge University Press. 1993.

Gottfried, Byron S. Theory & Problems of Programming with C (Schaum's Outline Series). McGraw-Hill Inc., Toronto. 1990.

Heiserman, David L. Programming in BASIC for the IBM Personal Computer. Prentice-Hall, NJ. 1984.

Hofstadter, Douglas R. Godel, Escher, Bach: An Eternal Golden Braid. Vintage Books, New York. 1979.

Hofstadter, Douglas R. Metamagical Themas: Questing for the Essence of Mind and Pattern. Basic Books, New York. 1985.

Holt, R.C. and Hume, J.N.P. Fundamentals of Structured Programming using FORTRAN with SF/K and WATFIV-S. Reston Publishing, Reston VA. 1977.

Holt, R.C. and Hume, J.N.P. Programming Standard Pascal. Reston Publishing, Reston VA. 1980.

Hume, J.N.P. Turing Tutorial Guide. Third Edition. Holt Software Associates, Toronto. 1989.

*Hume, J.N.P. and Holt, R. C. Introduction to Computer Science: Turing Programming Language. 2nd Edition, Holt Software Associates, Toronto. 1990.

Hume, J.N.P. Problem Solving and Programming in Turing. First Edition. Holt Software Associates, Toronto. 1993.

Hume, J.N.P. and Stephson, C. Programming Concepts in Java. Holt Software Associates, Toronto. 1998.

Hume, J.N.P. and Stephson, C. Introduction to Programming in Java. Holt Software Associates, Toronto. 2000.

International Computer Solving Contest: Senior Division Problems: 1981- 1991. ICPSC, 1991.

Kaye, Brian H. A Random Walk through Fractal Dimensions. VCH Publishers, New York. 1989.

Kernighan, Brian W. & Ritchie, Dennis M. The C Programming Language (2nd Edition). Prentice Hall, Englewood Cliffs NJ. 1988.

*Kochan, Stephen G. Programming in C. Hayden Book Company, New Jersey, 1983.

Koffman, Elliot B. Problem Solving and Structured Programming in Pascal. Addison-Wesley, Don Mills. 1981.

*Koffman, Elliot and Maxim, Bruce R.. Software Design and Data Structures in Turbo Pascal. Addison-Wesley, Don Mills. 1994.

Lewis, R.S. Fractals in Your Future: A Primer on Theory and Applications of Fractals and Chaos. published by the author, Sudbury. 1990.

McKeown, Patrick G. Structured Programming Using FORTRAN 77. Harcourt Brace Javanovich, Toronto. 1985.

Nickerson, Robert C. Fundamentals of Programming in BASIC: A Structured Approach (2nd Edition). Little, Brown & Company, Toronto. 1986.

Nissen, Walter I., Internet Forum: rec.puzzle, message dated 24 May 1995, Topic: "Integer Sequence"

Orilla, Lawrence S. Computers and Information (3rd Edition). McGraw-Hill, Toronto. 1986.

Pearson, Olen R. Programming with BASIC; A Practiacl Approach. McGraw-Hill, Toronto. 1986.

Peckham, Herbert & Ellis, Wade Jr & Lodi, Ed. Structured BASIC for the IBM PC. McGraw-Hill, Toronto. 1985.

Peitgen, Heinz-Otto & Saupe, Dietmar (editors). The Science of Fractal Images. Springer-Verlag, New York. 1988.

*Peitgen, Heinz-Otto, Jurgens, Hartmut & Saupe, Dietmar. Fractals for the Classroom: Part One: Introduction to Fractals and Chaos. Springer-Verlag, New York. 1991a.

*Peitgen, Heinz-Otto, Jurgens, Hartmut & Saupe, Dietmar. Fractals for the Classroom: Part Two: complex Systems & Mandelbrot Set. Springer-Verlag, New York. 1991b.

Peterson, Ivars. The Mathematical Tourist: Snapshots of Modern Mathematics. W.H. Freeman, New York. 1988.

Press, William H., Flannery, Brian P., Teukolsky, Saul A. & Vettering, William T. Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press. 1987.

Regan, Jim. Winning At Slot Machines. Citadel Press, NJ. 1985.

Rothenberg, Ronald I. BASIC Computing for Calculus. McGraw-Hill, Toronto. 1985.

Schroeder, Manfred. Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise. W.H. Freeman, New York. 1991.

*Sedgewick, Robert. Algorithms (2nd Edition). Addison-Wesley, Don Mills. 1988.

Solow, David. Thinking in Pascal: A Systematic Approach. Addison-Wesley, Don Mills. 1988.

Some Common Pascal Programs. Osborne/McGraw-Hill, Berkeley CA. 1982.

Spiegel, Murray R. Mathematical Handbook of Formulas and Tables (Schaum's Outline Series). McGraw-Hill Book Company, Toronto. 1968.

Stephenson, Chris (Editor). Turing Stuff: A Collection of Teacher Created Exercises, Projects & Quizzes. HSA, Toronto. 1994.

Tamburin, Henry. Reference Guide to Casino Gambling. Research Services Unlimited, Mobile. 1993.

*Tenenbaum, Aaron M. and Augenstein, Moshe J. Data Structures Using Pascal. Prentice-Hall, Englewood Cliffs NJ. 1981.

*Teukolsky, Roselyn. How to Prepare for the AP Computer Science Advanced Placement Examination. Barron's Educational Series, Inc. New York. 2001.

Turing, Alan Mathison. Mechanical Intelligence. Elsevier Science Pub, New York. 1992.

University of Toronto. Computing Insights 1995. (Summer computer course for high school students).

Weisenberg, Michael. Puzzled Programmers. Microsoft Press, Redwood WA. 1987.

Wilkinson, Russell M., Talsky, Wagner & Seung. Using the Computer: Concepts and Applications. Prentice-Hall, Scarborough. 1988.

www.cs.queensu.ca/home/cisc124/2001f/

www.cs.smsu.edu/~rcjudge/potw/probindex.html

www.cs.smsu.edu/contest.html

www.cs.toronto.edu

www.math.uwaterloo.ca/~ccc/

www.qucis.queensu.ca

www.undergrad.math.uwaterloo.ca/~cs130/

www.undergrad.math.uwaterloo.ca/~cs134/