NEW ZEALAND PROGRAMMING CONTEST

1994

PROBLEM A: 5 POINTS — HIGHEST

Read in a set of three digit numbers and output the highest number. The number may appear more than once on the file.

Input : Filename DATAA5 - note the 2 A's. There will be one number per record and no more than 100 records on the file. The end of file will be indicted by the number 999 which is not part of the input data.

Output : Screen with appropriate description.

Sample input :

```
123
304
253
926
034
587
999
```

Required output :

```
The highest number is 926
```

PROBLEM B: 5 POINTS — GRANTS

Students studying at Tony's Tertiary Tutory are entitled to a study grant if they are between the ages of 16 and 24 inclusive and have had less than 4 years tertiary training. Write a program to show whether a student is entitled to a grant. Your programm must supply satisfactory prompts. Those shown in the example are satisfactory.

Input : from screen, a two digit number for the age of the student and another number representing the years of tertiary study (this may be zero).

Output : to screen, either the message
The student is not entitled to a grant.
or the message
The student is entitled to a grant.

Sample input :

```
Enter age 23
Enter years of tertiary study 5
```

Required output :

The student is not entitled to a grant.

PROBLEM C : 5 POINTS — UPPER CASE

Convert a mixed case text line into upper case. The input may contain punctuation and alphanumeric characters.

Input : filename is DATAC5. The file will contain 2 records, the first with the line to be converted and the second with a hash indicating the end of file. The line to be converted will not exceed 80 characters.

Output : to screen

Sample input :

```
The quick Fox, always alerT is oN gUarD!
#
```

Required output :

```
THE QUICK FOX, ALWAYS ALERT, IS ON GUARD!
```

PROBLEM D : 5 POINTS — TIMES TABLE

Write a program that will request a single digit number from the screen and output the multiplication table for that number, up to 15 times.

Input : screen

Output : screen, the format may be seen from the sample below.

Sample input :

```
Enter a number from 0-9 of the times table required: 3
```

Required output :

```
1 times 3 = 3
2 times 3 = 6
    .
    .
    .
14 times 3 = 42
15 times 3 = 45
```

PROBLEM F : 10 POINTS — SEQUENCE

Generate the first 15 numbers of the sequence generated by starting with the two numbers N1 and N2, where the next number

is the sum of the previous two numbers.

Input : file name DATAF10. The file will contain 3 records, the first contains the number N1, the second N2 and the third the # (hash symbol) indicating end of input.

Output : to screen, each number is to separated by two spaces. The line should take approximately 280 characters.

Sample input :

```
1
3
#
```

Sample output :

```
1  3  4  7  11  18  29  47  76  123  199  322  521  843
1364
```

PROBLEM G : 10 POINTS CAESAR

A Caesar cypher is a simple code where one letter is converted to another by shifting it a fixed number of places in the alphabet. Foe example if the shift was 3 places, then a=d, b=e, c=f...w=z, x=a, y=b, z=c. Note that if the shift is 13, then the processes of encoding and decoding are the same.
Write a program which will accept one line of text, either coded or decoded and convert it to its opposite form. This means that the shift has to be 13. Upper and lower case must be maintained and all other characters unaltered.

Input : filename DATAG10. End of data is determined by a record containing a single # (hash symbol).

Output : screen

Sample input :

```
Advance B troop - to the front.
Nqinapr O gebhc.
#
```

Required output :

```
Nqinapr O gebhc - gb gur sebag.
Advance B troop.
```

PROBLEM H : 10 POINTS WORD COUNT

Write a program which will read in a line of text containing words which are separated by one or more spaces. A word is a set of characters starting with an upper case or lower case letter

and not containing a blank. Do not count punctuation.

Input : filename DATAH10. Records contain up to 100 characters. End of data is determined by a record containing a single # (hash symbol).

Output : Screen. Display the word count with an appropriate prompt.

Sample input :

```
Try this : Mr Smith-Jones
#                    .
```

Sample output :

```
The number of words is 4.
```

PROBLEM I : 10 POINTS — REVERSE NAMES

Write a program which accepts two names and displays them backwards and in reverse order.

Input : filename DATAI10. End of data is determined by a record containing a single # (hash symbol). Note that there could be trailing blanks.

Output : screen, the input data and the reverse data with appropriate explanations.

Sample input :

```
Fred Flintstone
#
```

Required output :

```
input data is : Fred Flintstone
reverse data is : enotstnilF derF
```

PROBLEM K : 20 POINTS — MAGIC GRID

Nine single digit numbers are arranged into a 3 by 3 grid. These numbers will be read in row-wise from a file with each row on a separate line. Originally the rows and columns added to the same number but one cell has been corrupted. Identify the number currently in the corrupted cell and print out the correct value for this cell.

Input : filename DATAK20. There are 3 records on the file, one for each row in the grid. End of data is determined by a record

containing a single # (hash symbol).

Output : screen.

Sample input :

```
448
316
952
#
```

Required output :

```
The value in the corrupted cell is 1.
The correct value is 7.
```

PROBLEM L : 20 POINTS — MODE

From a string of numbers which are represented in the format n.n or nn.n, find the most commonly occurring number(s).

Input : filename DATAL20, maximum number of records will be 100. End of data is determined by a record containing 99.9.

Output : screen, with appropriate prompts.

Sample input :

```
1.2
11.3
17.6
11.3
1.2
14.5
1.2
99.9
```

Required output :

```
The most common number is 1.2
```

PROBLEM M : 20 POINTS — ASCII SORT

Strings of 3 alphabetic characters are valued by adding the ASCII values of their original letters. Sort the strings into ascending order by value. Where the strings have the same value, they must appear in the output in alphabetic order.

Input : filename DATAM20, there will be a maximum of 100 records. End of data is determined by a record containing a single # (hash symbol).

Output : screen.

Sample input :

```
mba
reg
fzh
abm
mab
#
```

Required output :

```
abm
mab
mba
reg
fzh
```

PROBLEM N : 20 POINTS — CASH ANALYSIS

Write a program which will accept an amount between $1 and $100 and will generate the smallest number of notes and coins adding up to this amount. Coins in use are 5c, 10c, 20c, 50c, $1 and $2. Notes available are $5, $10, and $20.

Input : Filename DATAN20, input will be terminated by a record containing the value 00.00, which is not part of the input.

Output : screen. The presence of the plural 's' is required where necessary.

Sample input :

```
77.45
00.00
```

Required output :

```
3 x 20 dollar notes
1 x 10 dollar note
1 x 5 dollar note
1 x 2 dollar coin
2 x 20 cent coins
1 x 5 cent coin
```

Problem P Lightest Pairs (50 points)

The Annual Caber Tossing Championships are coming up, and this year the town of Shortbridge is determined to win the contest. This event is very popular, especially the Pairs Division, in which a pair of cabers, one spruce and one pine, are tied together and tossed as a unit. The performance of a particular pair of cabers in this event depends on the total weight of the cabers and also on the aerodynamic characteristics of the particular caber combination. The weight is easy to calculate, but the effects of air resistance can only be found by experiment. The usual selection procedure involves testing batches of pairs of cabers. Batches are selected so that the weights of the pairs in the batch are very similar. You have been hired by the selection committee to write a program to help select pairs for testing.

This year, to maximise the chance of winning a gold medal for Shortbridge, the selection committee has decided to cut down all large trees in the district; about 11,000 trees in total (it is known that there are no more than 6000 spruces and no more than 6000 pines). To pay for your services, Shortbridge has arranged to share expenses with several (smaller) surrounding towns. Pairs for a town are formed and ordered in the following way. All possible combinations of spruces and pines are considered. Pairs are sorted into order based on increasing pair weight. Pairs with the same weight are ordered based on increasing spruce number. Pairs with the same weight and spruce number are ordered based on increasing pine number. Spruce and pine numbers are simply the order in which caber weights appear in the input. Spruces and pines are numbered from 1. Your program is to make it possible to extract selected ranges from the overall pairs list. A range from X to Y selects the Xth to Yth pairs from the pairs list. Pairs are numbered from 1.

Input will be from a file PROBLEMP.DAT and will consist of details for a number of towns. The input for a town begins with a line that contains the town name; a string of no more than 40 characters (a town name of # denotes the end of input). Following that will be two lists of weights, the first for spruce cabers and the second for pines. Each list will begin with a line giving the number of items in the list, which will be followed by the weights, one per line. The weights will be integers in the range 1..500. Following the pines list are one or more ranges for which output is required. Each range consists of a lower number, at least 1, and an upper number which is at least the lower (the woodsmen's multiplication isn't very good, so both upper and lower may exceed the number of pairs). A line consisting of two zeroes signifies the end of the ranges required.

The output will consist of a series of town reports, one report for each town in the input. There will be a blank line separating town reports (but not following the final report). A town report begins with a line giving the town name (formatted as in the sample output below). This line is followed by a series of range reports, one for each range in the input. A range report begins with a line reporting the lower and upper limits of the range (formatted with a single leading blank and a single trailing blank as in the sample output below). This is followed by a series of lines detailing the caber pairs of the range, from lower to upper. Each line contains the spruce caber number, the pine caber number, and the combined weight. Each of the numbers is to be right justified in a field of 8 characters.

Problem P Lightest Pairs (Continued)

Example

INPUT

```
Blargsville
3
21
5
8
2
15
7
1 3
5 7
0 0
Shorttown
1
22
2
5
7
1 2
0 0
#
```

OUTPUT

```
Reports for town Blargsville
Pairs 1 to 3
        2       2       12
        3       2       15
        2       1       20
Pairs 5 to 7
        1       2       28
        1       1       36

Reports for town Shorttown
Pairs 1 to 2
        1       1       27
        1       2       29
```

Problem U Minimal T-Codes (100 points)

When a file consisting of characters is transmitted electronically, each character is replaced by a code, drawn from a Code Set. The ASCII code is frequently used, but is somewhat wasteful as there are 256 8-bit codes, while usually there are many fewer different characters in the set, and it is inefficient to use equal length codes for both frequent and infrequent characters. Variable-length codes can be used provided the Code Set is prefix-free, ie. no code is an extension of another code. A very desirable property of a Code Set is self-synchronisation - if a bit is missed out of the transmitted bit stream, only one or two characters should be confused. With ASCII, for example, a missing bit in one character completely changes the message from that point on; the first bit in each character becomes the last bit of the previous character.

One way of designing a prefix-free self-synchronising variable-length code is by using the T-code augmentation process. From a given prefix-free Code Set, one code is extracted, and a new Code Set formed using the remaining codes together with all the original codes with the chosen code on the front. For example, starting from the Code Set {1, 01, 001} and using the code 01 as the chosen code, the new Code Set {1, 001, 011, 0101, 01001} will be formed. This process forms prefix-free Code Sets that have excellent self-synchronising properties. If the starting set is {0, 1} and the augmentation process is used recursively, with one of the shortest codes in the set being used as the chosen code at each stage, then a minimal T-code set is formed. For example, {00, 01, 11, 100, 101} is the minimal T-code set formed by using first 0 then 1 as the chosen code. If 1 is chosen first, then 0, we get the minimal T-code set {10, 11, 00, 010, 011}. These are the only possible minimal T-code sets of order 2.

It turns out that the longest codes in minimal T-code sets have interesting properties; they are "incompressible", for example. The example above shows that there are 4 of these codes of length 3; {100, 101, 010, 011}. There are none of length 4, because the next stage of augmentation must add a code of length 2 to form the next longest code-words. However it is easy to see that there must be 12 of length 5. Your task is to write a program that will find if a given string of 0's and 1's is a longest code-word in some minimal T-code set.

Input will be from a file called PROBLEMU.DAT and will consist of a series of lines, each line containing a number string of 0's and 1's, of length <= 70 characters. The file will be terminated by a line consisting of a single 0.

Output will consist of a series of lines, one for each line of the input. Each line will consist of either the word "Yes" or "No", depending on whether the string is a longest code-word in some minimal T-code set or not.

Example

INPUT

```
011
1100101000001011100011110110100011100 11
1110101011110110110010110 11
1001011010110111001100010000110011111100001011101000100100111110000 11
1101100011110101001000101111001111100011110010100100111011000110
0
```

OUTPUT

```
Yes
Yes
No
Yes
No
```

Problem V Matching Moas (100 points)

You have been approached by the Blarg Toy company to assist them in developing a puzzle. The puzzle consists of a number of square cards. Each card has half of a picture of a moa on each of its four edges---either the upper (head) part of the moa, or the lower (tail part). Several different moas feature. When adjacent squares have the upper part of a moa on one edge and the lower part of the same moa on the adjoining edge of the neighbouring square then a match is said to have occurred; otherwise a mismatch has occurred. Blarg have asked you to write a program to solve puzzles. The input to the program will consist of one or more puzzles, each of which gives the values of M and N and MxN card specifications. For each puzzle, your program must print a legal MxN arrangement of the squares.

Input will be from a file named PROBLEMV.DAT and will consist of one or more puzzles to be solved. Each puzzle will begin with a line containing two integers: the number of rows M and columns N of the rectangle into which the squares must be arranged (1<=M<=20, 1<=N<=20). Values for M and N of 0 signify the end of input. Following the rectangle dimensions are M x N lines, each of which contains the details of a card. A card is defined by the pictures on its 4 edges. Each edge definition consists of a moa identifier (in the range 1 to MxN) followed by a H (for head) or a T (for tail). Edge definitions are separated by exactly one space. The edge definitions are given for the ``standard" orientation of the card, in top, right, bottom, left order. The first card in a puzzle is card 1, the second card 2, and so on.

Output will consist of one puzzle solution for each puzzle in the input. A puzzle solution is either a line containing the string "No solutions exist" or details of a layout that solves the puzzle. A layout consists of M rows of N card descriptions. A card description consists of the string I/J, where I is the card number (as defined above), and J is the number of 90 degree clockwise rotations of the card from the orientation given in the input. J is in the range 0 to 3. Card descriptions on the same line are separated by single blanks. Puzzle solutions are separated by a blank line; there should be no blank line after the last solution.

If there is more than one solution to a puzzle, then solutions should be compared on the basis of the card numbers in each position. The solution selected should have the lowest card number in row 1, column 1. If the same card is in the position in two solutions, then the one with the lowest number of right rotations for row 1, column 1 should be selected. If this number is the same for two solutions then the card in row 1, column 2 should be considered, and so on along each row until a difference is found.

Example

INPUT

```
2 2
2T 4H 1H 3T
4T 1T 3H 2H
1T 2T 3H 4H
1T 4H 3H 2T
1 2
1H 2H 3H 1H
4T 2H 3H 4T
0 0
```

OUTPUT

```
1/3 3/3
4/2 2/0

No solutions exist.
```

Problem W Playfair Code (100 points)

A top-secret message has been discovered by the British Ministry of War, which probably reveals the presence of one hundred tons of gold, buried around 1915 by a Home Guard unit who have all since died. The message was found in a pile of papers which have been in the staff club since the First World War. Unfortunately it is encoded using the Playfair code, and no-one knows the key, although it is sure to be four letters long (the commanding general appeared to know nothing other than 4-letter words). It is also sure that the message contains the words GOLD and WALACE (the name of the commanding general) You have been given the job of trying to decipher this message.

The Playfair code arranges the 25 letters of the alphabet (excluding Z) in a 5x5 square. The squares in the top left have the code word written in them, then the remaining letters of the alphabet are put into the square in order. For example, if the code word is "DAMN", the square will look like:

```
D A M N B
C E F G H
I J K L O
P Q R S T
U V W X Y
```

A message is encoded by first replacing all blanks by J, all Z's by S, and making all double letters into single letters. The message is then split into pairs of letters, and each pair encoded by finding a subrectangle of the Playfair table of which these letters form a pair of corners, and taking the opposite corners. For example, the pair OQ mark out the corners of a rectangle whose opposite corners are JT. The encoding of OQ is thus JT, the encoding of QO is TJ, the encoding of JT is OQ, and the encoding of TJ is QO. Note that the first letter of the coded pair is in the same row as the first letter of the original pair. If the letters are in the same column or row, the rectangle is deemed to have width or height 1 in the right or bottom direction, and the table is circular at its right and bottom edges; thus LS is encoded by OT, YH by UC and VX by AN. One advantage of this scheme is that the encoding and decoding processes are identical, except that when decoding the same-row or same-column case, it is necessary to go up or to the left. It is clear that knowledge of the key is crucial, although the encoding of most messages with the key "STOP", for example, will be very similar to the encoding of the same message using the key "SPOT".

Input will be from a file called PROBLEMW.DAT and will consist of a single code message, written as a number of lines of characters of at most 60 characters each (total number of characters is less than 500). Each line will have an even number of characters in it. Only uppercase letters will be used. The file will be terminated by a line containing a single #.

Output will be a set of lines, each line containing a 4-letter code word with the property that the decoding of the message using this code word has both the word GOLD and the word WALACE, both words being preceded by a blank (or the beginning of the message) and followed by a blank (or the end of the message). You must list all such code words, in alphabetic order; if there are no code words, write the message "No valid code words".

Examples

INPUT (Remember the input will contain only one code message)

```
VMJNIJOENQOQOJLETSAI
#
VMJNIJEBXYOJHLIN
#
FCXUNTIOBKMOJNXISMNDOSGTWPQKXPONTQAFJMWPPAOLTRTHOXBAIYAGCVQD
GFBPBGONTQAFELBHFAEQXPGFOLKHOCLDOAOKGBELAFHBYDOCELBPBGOMKYWY
WYABJLKREOUBHPFYIODSPKOGQEJOGFOKUBJAOATDOCYDOIGIBA
#
```

OUTPUT

```
DAMN

No valid code words

HIDE
```

Problem X Go Fish (100 points)

"Go fish" is a children's card game for 2 to 5 players, in which the object is to collect the most books, where a book is four cards that have the same face value. A standard deck is shuffled and dealt face down to the players, with the number of cards each player receives depending on the number of players (with 2 players each receives 7 cards, with 3 players each receives 6, and with 4 or 5 players each receives 5). The first card is dealt to the player (P) to the left of the dealer, the next to the player to the left of P, and so on until each player has the correct number. The dealer receives the last card. The rest of the deck is then placed face down on the middle of the table.

P starts the game by asking some other player (say Q) for all of their X cards, where X is the face value of one of the cards that P has in their hand. If Q has one or more X cards, they must give them all to P, and P then chooses another player to ask for X or asks any player for any of the other cards in P's hand. Sooner or later the P asks a player (say R) for a rank (card type) that R does not have. At this point R tells P to "Go fish", whereupon P picks up the card on top of the deck. The player to the left of P then begins asking for cards and so the game continues. If a player runs out of cards at any time during their turn, or finds they have asked every other player for every card they have in their hand, then their turn finishes and they draw a card from the top of the deck.

Whenever a player has a book in their hand (a book is four cards of the same value) they remove the cards from their hand. The winner is the player who has the most books when the game ends. The game continues until all books have been collected.

Write a program that will simulate playing this game. Remember that a standard deck (or pack) of cards contains 52 cards. These are divided into 4 suits---Spades, Hearts, Diamonds and Clubs. Within each suit there are 13 ranks---Ace (A), 2--9, Ten (T), Jack (J), Queen (Q) and King (K).

Each player is to follow the following strategy. First they consider all ranks which they know another player to possess. Consider the following situation. Player P asks player Q for all of her Kings. Q has no Kings so tells P to "Go fish". P draws a card, Q has her turn and then the turn passes to R, who does happen to have a King in their hand. R heard P asking Q for Kings, so knows that P has one or more Kings, so can ask P for Kings knowing that the request will be successful. If a player possesses two or more ranks that they know other players to possess then they should deal with them in the order in which they found out. Note, however, that once R has obtained P's Kings, then all players must delete the knowledge "P has a King", and insert the knowledge "R has Kings". This information is then the "newest" information about the whereabouts of ranks.

Having dealt with all cases where other players are known to have certain ranks, a player must then proceed to guess which players have which cards. The guessing strategy is as follows. The player (say P) selects one rank at a time. After asking all other players for that rank, or after forming a book in that rank, they then consider another rank. The rank asked for is the one that has been in P's hand the longest since the last time P asked any other player for a card of that rank. When a player picks up a card, it becomes the "newest" card in the hand, unless there is already card of that rank in the hand, in which case the new card simply "joins" the card already there, acquiring its "age". Assume a player picks up their cards in the reverse order to the one in which they received the cards from the dealer. For example, for the first example deck given, the dealer's hand, from "oldest" to "youngest", will be `D4 HJ (D5,S5) S6 CA`. The first rank the dealer asks for is 4 (successfully). This rank then becomes the "newest" rank in the dealer's hand, but the `J` which the dealer draws would become "newer" still (except that there is already a J in the dealer's hand).

When choosing a player to ask for cards of a certain rank, the players that have not been asked for that rank in the current turn are considered. Of those players, the one chosen is the who has not been asked by P (for a card of any rank) for the longest time (if two or more players have never been asked the one chosen is the one reached first by travelling clockwise from P around the group of players). Guesses continue until P runs out of cards, P is told to go fish, or P has asked every other player for every rank P currently holds. After the last guess, P takes the top-most card in the deck (if any cards remain in it).

Problem X Go Fish (Continued)

Input will be from a file PROBLEMX.DAT and will consist of a number of games. Each game begins with a line containing a number of players---an integer in the range 2 to 5. Following this will be the deck with which the game is played. Each deck will give the cards in order as they would be dealt (that is in the example deck below, ...) Decks will occupy 4 lines with 13 cards on each. The designation of each card will be the suit (S, H, D, C) followed by the rank (A, 2--9, T, J, Q, K). There will be exactly one space between cards. The file will be terminated by a game that specifies 0 players.

Output will consist of a series of lines, one for each game in the input. Each line will consist of n integers, where n is the number of players. Each integer is the number of books gained by one of the players. The first number is for the player to the left of the dealer, and so on clockwise around the circle of player. The last number is the number of books won by the dealer. Each number is right justified in a field of width 3.

Example

INPUT

```
2
HA H3 H4 CA SK S5 C5 S6 C4 D5 H7 HJ HQ
D4 D7 SJ DT H6 S9 CT HK C8 C9 D6 CJ C6
S8 D8 C2 S2 S3 C7 H5 DJ S4 DQ DK D9 D3
H9 DA SA CK CQ C3 HT SQ H8 S7 ST H2 D2
3
H4 H9 DJ C6 CJ DQ S8 D9 H6 CQ S7 C8 S3
DA C4 C7 S6 SQ HK C5 HJ SJ HQ DT H7 S4
S9 CA HT H3 SK CT H8 HA D8 D5 DK D6 C9
D3 H5 CK D4 SA C2 D2 C3 D7 S5 ST S2 H2
0
```

OUTPUT

```
  6  7
  4  7  2
```

IMPORTANT

Changes to the questions

Problem C

At the end of the first paragraph, add the sentence :

Non-letters must remain on the line unaltered.

Problem H

The first line should read:

Write a program which will read in a line of text containing words which are separated by one or more spaces, and count the words on the line.

Problem W

The last line of the third example is wrong - the last two characters are incorrect. The line should read:

```
WYABJLKREOUBHPFYIODSPKOGQEJOGFOKUBJAOATDOCYDOIGIBA
```