The 2002 British Informatics Olympiad
Sample paper

Time allowed: 3 hours

Instructions

You should write a program for part (a) of each question, and produce written answers to the remaining parts. Programs may be used to help produce the answers to these written questions but are not always necessary.

You may use a calculator and the on-line help that your programming language provides, and you should have a pen, some blank paper, and a blank floppy disk on which to save your programs. You must not use any other material such as disks, files on a computer network, books or other written information.

Mark the first page used for your written answers with **your name, age in years** and **school/college**. Number all pages in order if you use more than one sheet. All of your computer programs should display your name and school/college when they are run, and the floppy disk you use to submit the programs should also show your name and school/college.

For your programs to be marked, the source code must be saved, along with executables if your language includes a compiler; this includes programs used to help answer written questions. You must clearly indicate the name given to each program on your answer sheet(s).

Sample runs are given for parts 1(a), 2(a) and 3(a). **`Bold text`** indicates output from the program, and `normal text` shows data that has been entered. The output format of your programs should follow the 'sample run' examples. Your programs should take less than two minutes of processing time for each test.

Attempt as many questions as you can. Do not worry if you are unable to finish this paper in the time available. Marks allocated to each part of a question are shown in square brackets next to the questions. Partial solutions (such as programs that only get some of the test cases correct within the time limit, or partly completed written answers) may get partial marks. Questions can be answered in any order, and you may answer the written questions without attempting the programming parts.

Question 1

Eenie, Meenie, Mainee, Mo!

A group of friends are standing in a circle, playing a game. They proceed around the circle, chanting a rhyme and pointing at the next person (clockwise) on each word. When the rhyme finishes whoever is being pointed at leaves the circle. They then start again, pointing where they left off, and the game continues until only one person is left.

For example, suppose there are six friends (numbered 1 to 6) standing around the circle and the rhyme is "Eenee, Meenee, Mainee, Mo!". The first person to leave the circle is number 4, followed by 2 then 1, 3 and 6. Number 5 is left.

1 (a)
[24 marks]

Write a program that inputs two numbers (each between 1 and 1000 inclusive), the first being *n*, the number of friends, and the second the number of words in the rhyme. The output should be the number of the last person left.

You should assume that the friends are numbered (clockwise) from 1 to *n*, and that the count begins at person 1.

Sample run

```
Number of friends: 7
Words in rhyme: 3

Number 4 is left
```

1 (b)
[2 marks]

12 friends stand in a circle and use a rhyme with 5 words. In what order do they leave the circle?

1 (c)
[4 marks]

100 friends stand in a circle and play the game twice using the same rhyme. In the first game they count clockwise but for the second they count anticlockwise (in each game the count begins with person 1). The same person is left each time. What is the smallest number of words that might be in the rhyme?

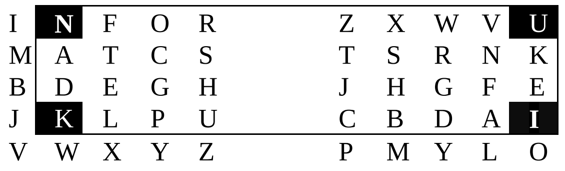
Question 2

Playfair Cipher

The *playfair cipher* is a way of encoding text. Your task in this question is to write a program that uses this method to encode and decode secret words. The code uses two 5x5 grids, each containing a jumbled alphabet (without the letter *Q*). To encode a word it is split into pairs of letters and each pair is encoded separately. [If the word has an odd number of letters an *X* is added to the end of the word before encoding]. For each pair of letters the first one is found in the left grid and the second in the right grid.

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

- If the two letters are on the same row, each one is replaced by the one to its right (wrapping around to the start of the same row if necessary). For example, *CR* would be replaced by *SN*, and *HE* would be replaced by *BJ*.
- If the two letters are in different rows, imagine they are two corners of a rectangle. They should be replaced by the two other corners in the rectangle. For example, *SE* is replaced by *HK*, and *KU* by *NI*.



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

So, the word *SECRET* becomes *HKSNTJ*, and *HELLO* becomes *BJXARW* (remember *X* was added).

To make it easier to remember the grids, they are each made from a key-word. To generate the left grid the key-word is written into the grid (top row first, starting at the left). In the example *INFORMATICS* has been used. Note that, if we come across a letter more than once (such as the *I*) we only use it the first time. The grid is then filled with the rest of the alphabet (in order), ignoring letters that have already been used.

The right grid is generated in the same way, but backwards, starting at the right of the bottom row. In the example the word *OLYMPIAD* is used.

2 (a)
[30 marks]

Write an interactive program to encode and decode secret words. Your program should do the following:

1. First, read in two lines. On each line will be a single word. The first line contains the key-word for the left grid and the second line contains the key-word for the right grid.
2. It should then output the two grids side by side.
3. Until your program terminates it should repeatedly wait for input and then:
   - If you receive the letter *E* you should input a word, encode it and print out the result.
   - If you receive the letter *D* you should input a word, decode it and print out the result. [You are expected to work out how to decode words.] No decoded word should end with the letter *X*
   - If you receive the letter *Q* you should terminate.
   - Ignore any other input.

Sample run

```
ROMULUS
REMUS

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

E
PLAYFAIR
CAOPKGZG

D
XGTJRO
GRIDS

Q
```

NB: All input words (including key-words) will have between 1 and 25 (inclusive) uppercase letters and will never include the letter *Q*.

2(b)
[3 marks]

In the sample run the letter *A* (in the word *PLAYFAIR*) was encoded to both *O* and *G*. In how many different ways can *A* be encoded?

2(c)
[4 marks]

Two pairs of grids are equivalent if the encoding of all words using the first pair is the same as the encodings using the second pair. How many equivalent pairs of grids are there to the *ROMULUS* and *REMUS* pair?

Question 3

A Space Oddity

Somewhere deep in space the computer AHN has gone crazy and is refusing to open the pod-bay doors between the space station it controls and a pod containing several astronauts. Since they cannot dock the pod with the space station, the astronauts will be forced to take a difficult space walk between the pod and the emergency airlock on the station.

Your task in this question is to get the astronauts safely onboard the station, as quickly as possible. Help them stop AHN before it is too late.

Fortunately each astronaut has their own space-suit. Unfortunately they only have one portable oxygen-pack between them, which the suits require to work. This pack can be used simultaneously (if necessary) by two astronauts travelling together, but it cannot provide oxygen for three or more at the same time. Since there is only one pack it will have to be taken back and forth until all the astronauts are safely onboard.

It is a difficult space walk and abilities of the astronauts vary, so their times to travel between the pod and the space station also vary. Two astronauts travelling together move at the speed of the slowest astronaut, since they are kept together by the oxygen-pack.

For example, suppose *A* takes 1 minute, *B* takes 2 minutes, *C* takes 4 minutes and *D* takes 5 minutes. The following combination of walks gets the astronauts onto the space station in 13 minutes. *A* and *D* travel to the station (5 minutes), *A* travels back to the pod (1 minute), *A* and *C* travel to the station (4 minutes), *A* travels back to the pod (1 minute) and finally *A* and *B* travel to the station (2 minutes). It can be done faster!

3(a)
[30 marks]

Write a program that calculates the fastest way of getting the astronauts onto the space station. Your program should first input a single number *n*, 1≤*n*≤8, indicating the number of astronauts. The second line will consist of *n* sorted integers (between 1 and 100), giving the times taken by each astronaut.

Sample run

```
4
1 2 4 5
12
```

Your output should consist of a line containing the minimum amount of time needed to get the astronauts onto the space station.

3(b)
[3 marks]

How do you get the astronauts in the example over to the space station in 12 minutes?

3(c)
[bonus]

Explain *briefly* the best possible algorithm (method) you can think of for solving the problem. (Do not worry if you do not know how to program your algorithm). If the input values could be greater, how large a problem would this method be able to solve?

Total marks: 100 (plus bonus).

End of BIO 2002 Sample paper