British Informatics Olympiad

Time allowed: 3 hours

The 2012 British Informatics Olympiad

LIONHEAD STUDIOS

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. You should have a pen, some blank paper, and an empty USB stick (or other storage device) 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. You may not communicate with anyone, other than the person invigilating this paper.

Mark the first page of 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 storage device 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. Where multiple items of input appear on the same line they should be separated by a single space. The output format of your programs should follow the 'sample run' examples. Your programs should take less than *5 seconds* 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.

Hints

- If you can only see how to solve part of a problem it is worth writing a program that solves that part. We want to give you marks and questions are evaluated using multiple tests of differing difficulty. ***Remember, partial solutions may get partial marks.***
- Question 2 is an implementation challenge and question 3 is a problem solving challenge.
- Most written questions can be solved by hand without solving the programming parts.
- Do not forget to indicate the name given to your programs on your answer sheet(s).

Question 1: *Distinct Prime Factorisation*

A *prime number* is a whole number, greater than 1, that can only be divided by itself and the number 1. Every integer greater than 1 can be uniquely expressed as the product of prime numbers (ignoring re-ordering those numbers). This is called the *prime factorisation* of the number.

For example:

- $100 = 2 \times 2 \times 5 \times 5$
- $101 = 101$ (since 101 is a prime number)

In this question we are interested in the product of the distinct prime factors of a given number; in other words each number in the prime factorisation is to be used only once.

For example:

- Since $100 = 2 \times 2 \times 5 \times 5$ the product we require is 10 (i.e. 2×5)

1(a) [24 marks]

Write a program which reads in a single integer n ($1 < n < 1{,}000{,}000$) and outputs a single integer, the product of the distinct prime factors of n.

Sample run 1

```
100
10
```

Sample run 2

```
101
101
```

1(b) [2 marks]

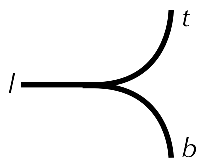
Which are the 10 lowest numbers having 10 as the product of their distinct prime factors?

1(c) [4 marks]

Which product of distinct prime factors, for n between 1 and 1,000,000, occurs the most frequently?

Question 2: ***On the Right Track***

Consider a railway that consists of lengths of track that are connected together by simple *points*.



A train entering a point along one of the curved pieces of track must exit along the straight portion; i.e. a train entering from *t* or *b* will exit via *l*. A train entering via *l* will exit via *t* or *b* depending on the point.

We will consider two types of points:

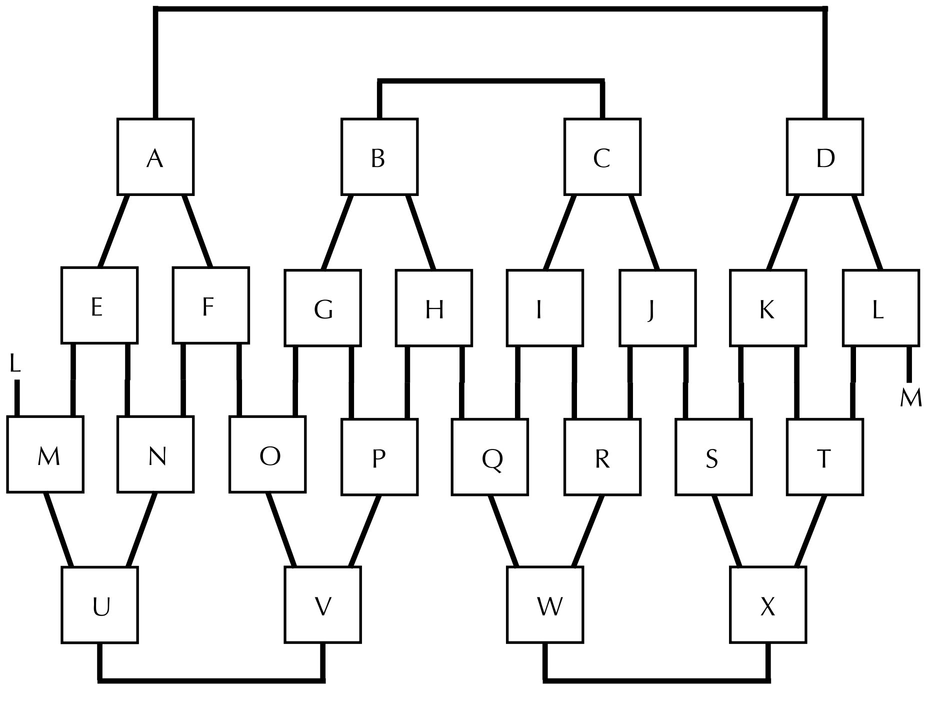
On a *lazy point*, entering from a curved portion sets the point so that, until the point is set again, a train entering from the straight portion will exit on the set curved portion. E.g. If a train enters the track from *t* (hence leaving via *l*) whenever it then enters the point from *l* it will exit from *t*, until the point's setting is altered by a train entering from *b*.

On a *flip-flop point*, every time a train has entered from the straight portion and exited on a curved portion, the point will change so that the next time a train enters the straight portion it will exit on the other curved track. Entering on the curved portion does not affect the point.

In this question we will be modelling a train moving along the railway network shown below. Each point is represented by a labelled square; each of which has one edge connected to a single piece of track (the straight portion for that point) and another edge connected to two pieces of track (the curved portions for that point.)

We will write *x* → *y* to indicate the train is between points *x* and *y*, moving towards *y*.

Initially each point is set to the left in the diagram. E.g. If the train starts at *A* → *E*, and all the points are *lazy,* it would pass through points in the order E, M, U, V, O, F, A, …



NB: Points L and M are connected together by track, but for clarity this is not shown above.

2(a) [23 marks]

Write a program that models a train moving along the railway network.

Your program should read in three lines of input. The first line of input will contain six different uppercase letters, indicating that the corresponding points are *flip-flop* points; all other points will be *lazy* points. The second line of input will contain two letters, *x* then *y*, indicating the train is at $x \rightarrow y$. The final line of input will be an integer *n*, ($1 \leq n < 10{,}000$), giving the number of points the train is to pass in the simulation.

You should output two letters, the first indicating the last point the train passed and the second indicating the next point the train will pass through.

Letters in the input and output should *not* be separated by spaces.

Sample run 1

```
GHIJKL
AE
6
FA
```

Sample run 2

```
GHIJKL
AE
100
VP
```

2(b) [2 marks]

Suppose all the points are *lazy* and the train is at $P \rightarrow V$. Write out the sequence of points the train passes up to and including when it passes P for the first time.

2(c) [6 marks]

Suppose, other than the layout of the railway, you know nothing about the points (either their type or how they are set) and that the train starts at $P \rightarrow V$. What can you say about the train's position after it has passed 1,000,000,000,000,000,000 points? Justify your answer.

2(d) [4 marks]

A red and a blue train have been placed on different parts of the railway and will move simultaneously (i.e. whenever one train passes a point the other train will also pass a point). A safety system will kick-in, freezing all movement, if two trains attempt to pass the same point at the same time or try to occupy the same track between points at the same time. It is possible to place the trains on the railway (all points being lazy and initially set to the left) so that the safety system never kicks-in — in how many ways?

Question 3: ***Number Ladder***

A number can be transformed to another number if the letters in the digits of the first number are the same as those in the second number with at most 5 letters (in total) being added or deleted. Individual digits are spelt ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE and ZERO.

For example:

- 61 can be transformed to 26 since the letters SIXONE, with 4 changes (N and E deleted, and T and W added), can then be arranged to TWOSIX. Note that the order of the letters does not matter.

A *number ladder* is a sequence of transformations that changes one number to a *different* number. Between any two adjacent numbers on the ladder a valid transformation takes place. No number appears more than once on a ladder. The *size* of a ladder is the number of transformations that takes place.

For example:

- 1, 90, 610 is a number ladder (of size 2) since 1 can be transformed to 90 and 90 can be transformed to 610. There is no smaller number ladder that starts with 1 and finishes with 610 since 1 cannot be transformed to 610.

3(a) [23 marks]

Write a program to determine the size of smallest number ladders.

Your program should read in three lines, each containing two integers s then f, ($1 \leq s < f \leq 999$). You should output three numbers (one per line), the i^{th} of which giving the size of the smallest number ladder (that only uses numbers between 1 and 999 inclusive) going from s to f on the i^{th} line of the input.

You will only score points if all three numbers in your output are correct.

Sample run

```
26 61
1 90
1 610
1
1
2
```

3(b) [2 marks]

How many integers, excluding 0, can be transformed (directly) to ZERO?

3(c) [4 marks]

Suppose that, for each possible start and finish (between 1 and 999 inclusive) we have calculated the smallest possible number ladder. What is the largest size of these ladders? How many ladders have this size?

3(d) [6 marks]

Does a number ladder exist to change any integer, however large, into any other integer with the same number of digits? Justify your answer.

Total Marks: 100 End of BIO 2012 Round One paper