British Informatics Olympiad

Time allowed: 3 hours

The 2005 British Informatics Olympiad

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 a blank floppy disk (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 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 **10 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: ***Fractions***

Given a decimal fraction we can write it as a proper fraction in its lowest form. Recall that a proper fraction $^a/_b$ is in its lowest form if it is not possible to divide a and b by a common factor.

For example, 0.25 is written as $^1/_4$ in its lowest form.

For the whole of this question, we are only interested in proper fractions in their lowest form.

1(a) [24 marks]

Write a program which, given a decimal fraction between 0.0001 and 0.9999 inclusive (and no more than four decimal places), outputs the equivalent proper fraction. The input will always start with a 0 before a decimal point, and all decimal fractions are exact (i.e. they are not just approximations to four decimal places).

Sample run

```
0.125
1 / 8
```

1(b) [2 marks]

In the fraction $^a/_b$ the lower number b is called the denominator. How many different denominators would be produced if we calculated proper fractions for all the decimal fractions (with no more than four decimal places) between 0.0001 and 0.9999?

1(c) [4 marks]

We can score a decimal fraction by multiplying together all of the digits in its equivalent proper fraction. For example, 0.04 has a score of 10 since its equivalent fraction is 1/25 and 1 * 2 * 5 = 10, but 0.05 has a score of 0 since its equivalent fraction is 1/20 and 1 * 2 * 0 = 0.

Which decimal fraction between 0.0001 and 0.9999 (with no more than four decimal places) has the highest score?

Question 2: *Turing Machine*

A *Turing Machine* is a simple type of programmable computer. It consists of a *tape* which can be read from and written to, and a set of *instructions* which tell it how to behave. The tape is a long line of cells, each of which can contain a `0` or a `1`. The machine is always positioned at a cell on the tape and is in one of several numbered *states*.

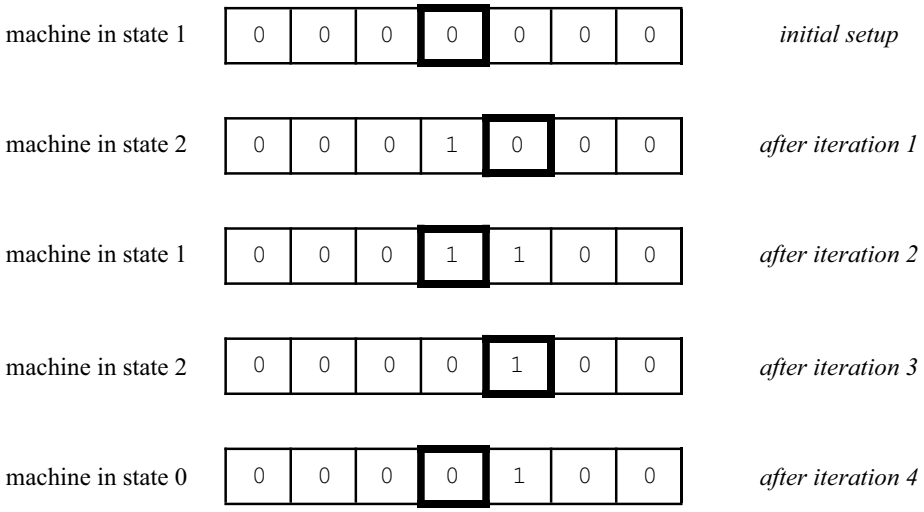
All instructions are all of the form "if we are in state *s* and there is an *r* in the current cell, write *w* in the current cell, then move one cell in direction *d* (left or right) on the tape and finally change into state *t*". For our simulation, each state (except state 0) will have two rules, one for what to do if there is a `0` in the current cell and another if there is a `1`. State 0 is special; it has no rules and if the simulation reaches this state it will terminate.

Our machine will always start off in state 1 and all of the cells on the tape will start off set to `0`. The tape is very long and we will never reach either end, whether we keep moving left or right.

Here is an example set of instructions:

Current State *(s)*	Current Value *(r)*	New Value *(w)*	Direction *(d)*	New State *(t)*
1	0	1	right	2
1	1	0	right	2
2	0	1	left	1
2	1	1	left	0

Here is how these instructions affect the tape (the highlighted box indicates the current position on the tape):



One instruction is processed on each iteration of the simulation until state 0 is reached. There are 4 iterations in the example simulation.

2(a) [25 marks]

Write a program to model a Turing Machine. Your program should first read in a line containing a single integer n ($1 \le n \le 3$) indicating the number of states (excluding state 0) for the machine.

The next n lines will consist of two triplets, each consisting of a single digit w (`0` or `1`), a letter d (`R` or `L`) and a single digit t (between 0 and n inclusive). The first triplet on the i^{th} of these lines is the instruction for when a `0` is read in state i; the second is for when a `1` is read in this state. The triplet wdt indicates that a w should be written, the machine should move in direction d (`R` for right and `L` for left), and that the new state should be t.

The final line will contain a single integer m ($1 \le m \le 1{,}000{,}000$) indicate the number of iterations of the simulation to run. You should run the simulation until this many iterations have taken place, or state 0 has been reached.

Your output should consist of two lines, the first containing the final values of 7 adjacent cells on the tape, the middle value showing the value for the final position on the tape. The second line should contain a single integer indicating the number of iterations that took place.

Sample run

```
2
1R2 0R2
1L1 1L0
10

0000100
4
```

2(b) [2 marks]

Suppose that, rather than starting with a tape containing just `0`s we start with one containing n (n>1) adjacent `1`s, followed (to the right) by a single `0`, followed (to the right) by m (m>1) adjacent `1`s. If we start the simulation at the leftmost `1` on the tape, what will be on the tape when the following program reaches state 0?

```
1R2 1R1
0L3 1R2
0L3 0L4
0R0 1L4
```

2(c) [3 marks]

A *busy beaver* is a set of instructions that will reach state 0 having left as many `1`s as possible on the tape. For a machine with 2 states (in addition to state 0) it is possible to leave 4 `1`s and terminate. Find this busy beaver.

2(d) [5 marks]

Suppose we have two Turing Machines, with potentially different sets of instructions, operating on the same tape at the same time. On each iteration, each machine executes one instruction. If both machines start an iteration on the same cell there may be conflicting instructions, so we may need to make a special rule. If the machines begin the simulation on adjacent cells will such a rule ever be needed? Justify your answer.

Question 3: *Movie Magic*

The Hollywood studio *Greenlight Casting Couch* is producing its latest movie extravaganza. Each actor has a certain number of scenes to film and, to stay in character, each actor's scenes are to be filmed in order. There is also a hierarchy amongst the cast and it is important that, at all times, no actor has filmed more scenes than a more senior actor. The movie, a post-modernist black & white fairytale told in flashback, has no scene containing more than one actor.

The following is an example schedule for 2 actors. The numbers indicate the order in which the scenes are being shot, the top row showing those scenes for the senior actor and the bottom row showing those for the junior actor. Note that numbers increase from left to right (each actor's scenes are filmed in order) and from top to bottom (at all times the senior actor has filmed more scenes).

```
1  2  4
3  5
```

This is only one of 5 possible schedules for this film. The other possible schedules are:

```
1  2  3      1  3  5      1  2  5      1  3  4
4  5         2  4         3  4         2  5
```

3(a) [25 marks]

Write a program which finds the number of possible schedules for a given movie.

The input will consist of two lines. The first will contain a single integer n ($1 \leq n \leq 8$) indicating the number of actors. The second line will contain n integers (each between 1 and 8 inclusive) indicating the number of scenes for each actor, in order of seniority (most senior first).

Sample run

```
2
3 2
5
```

Your output should consist of a line giving the number of possible schedules. Every test case will have an answer between 1 and 1,000,000 inclusive.

3(b) [3 marks]

Suppose the hierarchy condition was dropped (i.e. so the only condition is that each actor films their own scenes in order). If there are 2 actors, each with 2 scenes, how many possible schedules are there? How about if there are 3 actors, each with 3 scenes?

3(c) [4 marks]

The most senior actor's first scene must be the first scene shot in every schedule. For what movies must the most junior actor's final scene always be the last scene shot? Justify your answer.

3(d) [3 marks]

Support 3(a) was modified so the outputs had to be at least 1, but could be arbitrarily high. How many different inputs are valid for the modified question?

Total Marks: 100 **End of BIO 2005 Round One paper**