British Informatics Olympiad

Time allowed: 3 hours

The 2010 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. 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: ***Anagram Numbers***

An *anagram number* is a number that can be multiplied by at least one single digit number (**other than 1**) to become an anagram of itself. Any anagrams formed by multiplying an anagram number by a digit are said to be *generated* by that anagram number. Two numbers are anagrams of each other if they can both be formed by rearranging the same combination of digits.

For example:

- 1246878 is an anagram number; multiplying by 6 generates 7481268 or by 7 generates 8728146. These numbers all contain a single 1, 2, 4, 6, 7 and two 8s.
- 1246879 is not an anagram number.

1(a) [25 marks]

Write a program which reads in a single number (between 1 and 123456789 inclusive) and determines if it is an *anagram number*.

If the number is not an anagram number you should output the word **NO**. If it is an anagram number you should output each single digit it can be multiplied by to make an anagram of itself.

Sample run 1

```
123456789
2 4 5 7 8
```

Sample run 2

```
100
NO
```

1(b) [2 marks]

85247910 is generated by which anagram numbers?

1(c) [3 marks]

How many anagram numbers between 100,000 and 999,999 contain no duplicated digits?

Question 2: *Die Tipping* *NB: Die is the singular of dice*

A normal *die* is a cube where the faces are numbered 1 through 6 and the numbers are arranged so that opposite faces add up to 7. In this question we will move a die on an 11 by 11 grid. Each square on this grid is the same size as the faces of the die; i.e. the die fits exactly onto each square on the grid.

Imagine the die is placed on the grid so that the number 1 is uppermost and 6 is touching the grid. The die is now rotated so that 2 is towards the top of the grid and 5 is towards the bottom. This question uses a die that has 3 on the left and 4 on the right. We will call this the *default orientation* of the die.

A *move* is made by tipping the die over one of the edges of the face that touches the grid. This will move the die one square on the grid and change the face that touches the grid. If the die is tipped off the left side of the grid it should be placed at the corresponding position (i.e. with the same y co-ordinate) on the right side of the grid, and vice versa. Similarly for the top and bottom of the grid (with the x co-ordinate staying the same).

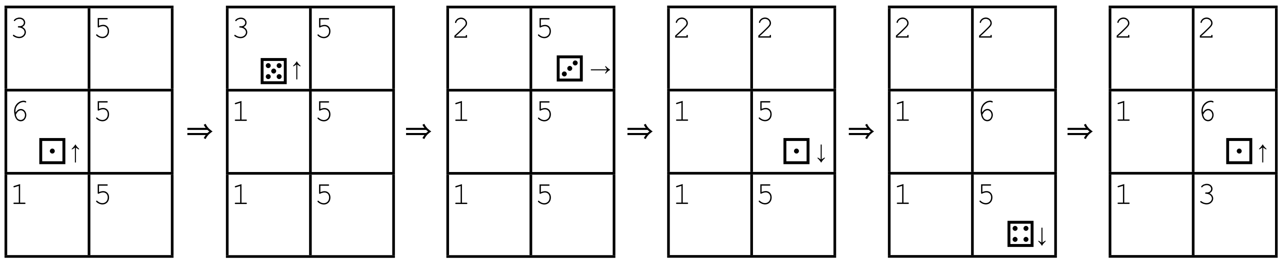
For example, suppose the die is in its default orientation in the middle of the grid. It now tips over the edge towards the bottom of the grid. The die moves one square closer to the bottom of the grid, and 5 now touches the grid. 1 is now towards the bottom of the grid, 6 the top, 2 will be uppermost and 3 and 4 will remain on the left and right respectively.

The die also has a *heading*, which is the direction on the grid it last moved.

The rules for moving the die are as follows:

1. Each square on the grid contains a number (1, 2, 3, 4, 5 or 6). Add the number on the current square to the value that is uppermost on the die. If this sum is greater than 6, subtract 6 from the sum.
2. Replace the number on the grid with this new value.
3. Depending on this value perform one of the following actions:
   - 1 or 6 – move one square according to the heading;
   - 2 – move one square in the direction 90° clockwise to the heading;
   - 3 or 4 – move one square in the opposite direction to the heading;
   - 5 – move one square in the direction 90° anti-clockwise to the heading.

In the following example part of the grid is shown at the end of each move. The numbers indicate the values written on the grid, the dots in squares indicate the value uppermost on the die (it starts in the default orientation) and the adjacent arrow indicates the current heading. A smaller than actual die is shown in the example for convenience.



2(a) [24 marks]

Write a program that models the die moving on the grid.

Your program should first read in a 3×3 grid, which will be in the form of three lines of three integers (between 1 and 6 inclusive). This 3×3 grid should be used as the centre section of the 11×11 grid for the simulation. The remaining squares of the grid will start with the value 1.

In each case the die begins in the centre of the 11×11 grid, in the default orientation, with a heading towards the top of the grid.

Until your program terminates you should repeatedly input an integer, and then:

- If you receive a positive integer *n* ($1 \le n \le 100$) you should make *n* moves.
- If you receive the number 0 your program should terminate immediately.
- Ignore any other input.

After each positive integer you should output the 3×3 grid surrounding the die. For each square on this grid that is outside the 11×11 grid you should output an **x**.

Sample run

```
1 3 5
1 6 5
1 1 5
1

111
135
115

1

111
251
151

3

221
161
131

0
```

2(b) [2 marks]

Suppose that the die is in the centre of the 11×11 grid, in the default orientation, with a heading towards the top of the grid. The numbers on the grid are such that it moves directly to and then over the right edge of the grid; i.e. on the sixth move it tips over the right edge. Considering only the squares it touches before tipping over the right edge, in how many different ways might they be numbered?

2(c) [4 marks]

Suppose that the die can be tipped in any direction on each move. How many sequences of 4 tips are there that finish with the die in its original position, but with a different face uppermost? How many such sequences are there taking 6 tips?

2(d) [5 marks]

Consider a sequence of squares occupied by the die as it travels. If, as the die continues to move, this sequence keeps repeating exactly and never changes, we will say that the die is *stuck in a loop*. Note that the die might have made several moves before reaching such a sequence.

Suppose that you are able to choose all the numbers of the board as well as the starting position, orientation and heading of die. Is it possible to select these values so that the die will never get stuck in a loop however long it moves? Justify your answer.

Question 3: *Juggl(ug)ing*

A cookery set contains several jugs of ***different*** sizes, each of which has a known capacity in oz (fluid ounces). There are no graduations on the jugs, so the only way to measure the liquid is by filling up a jug completely, pouring as much liquid as possible from one jug into another or by emptying a jug. A *step* consists of performing one of these three operations.

Initially all the jugs are empty. To measure *n* oz it is necessary to perform a sequence of steps and finish with exactly *n* oz in one of the jugs.

For example, suppose we have two jugs: jug A holds 3 oz and jug B holds 5 oz.

- If we fill up jug B and then pour as much as possible from jug B into jug A, we would have (after 2 steps) 3 oz in jug A and 2 oz in jug B. This is one way to measure 2 oz.
- If we now empty jug A, pour the 2 oz from jug B into jug A, fill jug B and finally pour as much as possible from jug B into jug A, we would finish (after 6 steps) with 3 oz in jug A and 4 oz in jug B. We have now measured 4 oz.

3(a) [23 marks]

Write a program that, given the capacities of several jugs, determines the shortest number of steps necessary to measure *n* oz.

The input will consist of two lines. The first line will contain two integers *j* ($1 \le j \le 3$) then *n* ($1 \le n \le 250$), indicating the number of jugs and the required amount to measure respectively. The second line will contain *j* integers, each between 1 and 250 inclusive, indicating the capacity of the jugs.

You should output a single integer giving the minimum number of steps necessary to measure *n* oz.

Your program will only be asked to measure amounts that are possible.

Sample run

```
2 4
3 5
6
```

3(b) [2 marks]

Given two jugs A and B, whose capacities are 3 oz and 8 oz respectively, show how to measure 2 oz in 4 steps.

3(c) [5 marks]

Two cookery sets that can measure the same values and no others are said to be *equivalent*. For example, the pair of jugs of capacity 5 oz and 9 oz, and the pair 2 oz and 9 oz can both measure every value from 1 oz to 9 oz. Ever cookery set is equivalent to itself and the order of the jugs in a set does not matter.

Consider the cookery set containing three jugs whose capacities are 6 oz, 18 oz and 20 oz. How many equivalent cookery sets are there that contain two jugs? How many are there that contain three jugs?

3(d) [5 marks]

Is it possible to find a cookery set, containing no 1oz capacity jugs, with a sequence of steps that leaves 1oz in all the jugs simultaneously? Justify your answer.

Total Marks: 100 End of BIO 2010 Round One paper