British Informatics Olympiad

LIONHEAD STUDIOS

The 2003 British Informatics Olympiad

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. 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 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 **30 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
ISBN

An **ISBN** (International Standard Book Number) is a ten digit code which uniquely identifies a book. The first nine digits represent the book and the last digit is used to make sure the ISBN is correct. To verify an ISBN you calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third ... all the way until you add 1 times the last digit. If the final number leaves no remainder when divided by 11 the code is a valid ISBN.

For example 0201103311 is a valid ISBN, since 10*0 + 9*2 + 8*0 + 7*1 + 6*1 + 5*0 + 4*3 + 3*3 + 2*1 + 1*1 = 55.

Each of the first nine digits can take a value between 0 and 9. Sometimes it is necessary to make the last digit equal to ten; this is done by writing the last digit as X. For example, 156881111X.

1 (a)
[24 marks]

Write a program that reads in a valid ISBN with a single missing digit, marked with a ?, and outputs the correct value for the missing digit.

Sample run

```
ISBN: 15688?111X
Missing Digit: 1
```

1 (b)
[2 marks]

Which of the following codes are valid ISBNs: 0972311900, 3540678654, 9514451570 and 013674409X?

1 (c)
[4 marks]

A valid ISBN has two of its digits swapped and the resulting code is 3201014525. What are the possible values of the valid ISBN?

Question 2
Waves

A river has burst its banks, creating a virtual lake. Pebbles are being thrown into this lake, causing waves to flow across its surface. Your task in this question is to model the waves. *(NB: We are not trying to build an accurate simulation of the real world; you should only implement the behaviours detailed in the question.)*

The surface of the lake is represented by an infinitely large grid of squares. The water at each square is at a depth, which is determined by the interaction of the waves. Before any pebbles are dropped, every square (except those on the original river banks) has a depth of 0. When displayed, each square will be represented as follows:

o	Square has a depth < 0
-	Square has a depth = 0
*	Square has a depth > 0
X	Square is part of the original river bank

When a pebble in dropped into the lake it causes a wave of raised water to spread out in a diamond pattern, getting larger every second. The depth of each square in the diamond is increased by 1 for a single second. This wave is followed, two seconds later, by a wave of lowered water (depth decreased by 1) which also spreads out as a diamond. Each pebble causes only two waves, the raised wave followed by the lowered wave.

For example, the effect of a pebble on the lake 0, 1, 2 and 3 seconds after it is dropped is:

```
-------     -------     -------     ---*---
-------     -------     ---*---     --*-*--
-------     ---*---     --*-*--     -*-o-*-
---*---     --*-*--     -*-o-*-     *-o-o-*
-------     ---*---     --*-*--     -*-o-*-
-------     -------     ---*---     --*-*--
-------     -------     -------     ---*---
```

The two river banks have fixed x co-ordinates and run for the entire length (top to bottom) of the lake with a width of a single square. When a wave reaches a river bank, rather than carrying on, it continues to grow but the part that hits the bank is reflected.

The early seconds of a wave hitting one of the river banks are as follows:

```
X------     X------     X--*---     X-*-*--     X*---*-
X------     X--*---     X-*-*--     X*---*-     X*----*
X--*---     X-*-*--     X*---*-     X*----*     X-*----
X-*-*--     X*---*-     X*----*     X-*----     X--*---
X--*---     X-*-*--     X*---*-     X*----*     X-*----
X------     X--*---     X-*-*--     X*---*-     X*----*
X------     X------     X--*---     X-*-*--     X-*--*-
```

NB: To make the picture clearer we are not showing the inner wave of lowered water.

When sections of waves meet they have no effect on the way each other behaves; In other words, at the next clock tick the waves will grow as though there was no encounter. Note however that their effects are combined on the lake. For example:

```
---------     ---------     ---------     ---*-----
---------     ---------     ---*-----     --*-*-*--
---------     ---*-----     --*-*-*--     -*-o-*-*-
---*-----     --*-*-*--     -*-o-*-*-     *-o-----*
--*-*-*--     -*-o-*-*-     *-o-----*     -o-*-o---
---*-----     --*-*-*--     -*-o-*-*-     *-o-----*
---------     ---*-----     --*-*-*--     -*-o-*-*-
---------     ---------     ---*-----     --*-*-*--
---------     ---------     ---------     ---*-----
```

2 (a)
[27 marks]

Write a program to simulate the waves:

Your program should first read in a single number *p* (1 ≤ *p* ≤ 5) indicating the number of pebbles. This will be followed by *p* lines, each containing three integers: the x, then y co-ordinate for the pebble (each between -500,000 and 500,000 inclusive) followed by the time *t* at which the pebble is dropped into the lake (1 ≤ *t* ≤ 500,000). The next line will contain two integers (each between -500,000 and 500,000 inclusive) giving the x co-ordinates of the two river banks. There will then follow a final line with a single integer *r* (1 ≤ *r* ≤ 500,000), the time at which you are to display the lake.

No two pebbles will be dropped at the same position at the same time. The two banks will have different x co-ordinates, and no pebble will be dropped on a bank.

You should output a 9 x 9 grid, centred on 0,0; the bottom left of your grid representing (-4, -4) and the top right (4,4). The grid should represent the state of the lake at time *r*.

Sample run

```
2
-3 0 1
0 0 2
4 100
4
--------X
-*------X
*-*-*---X
-o-*-*--X
o-----*-X
-o-*-*--X
*-*-*---X
-*------X
--------X
```

2 (b)
[3 marks]

The grid below shows the state of the lake at time 2 with an unspecified number of pebbles (with correspondingly unspecified locations and times). There are no other raised or lowered parts of the lake. How many different grids might this lead to at time 3, assuming no pebbles are dropped into the lake at that time?

```
-----
--*--
-*-*-
--*--
-----
```

2 (c)
[4 marks]

Two pebbles are dropped into a lake, with no river banks, at different locations at time 1. If you can choose the locations at which the pebbles are dropped, at time 100 what is the maximum number of raised squares that might exist? What is the minimum?

Question 3
New Order

Numbers are normally ordered by their magnitude, so `10` comes before `110` which comes before `1000`. In this question we will consider a different way of ordering the numbers. We are also going to restrict ourselves to numbers that only use the digits `0` and `1`.

The new order rule is:

> If a number is written with fewer *1*s than another number, the number with the fewest *1*s comes first. If both numbers are written with the same number of *1*s the number with the smallest magnitude comes first.

With this ordering, `1000` now comes before `110` (since it has fewer *1*s) but still comes after `10` (since it has a greater magnitude). The order of the 16 numbers that have no more than 4 digits is:

```
   0
   1
  10
 100
1000
  11
 101
 110
1001
1010
1100
 111
1011
1101
1110
1111
```

3 (a)
[21 marks]

In the new order, all the numbers with the same number of *1*s are grouped together. Write a program to find the *n*th number with exactly *m* *1*s.

Sample run

```
3 4
11011
```

The input will be a single line consisting of two integers. The first integer *n* ($1 \le n \le 1{,}000{,}000{,}000$) indicates you are to find the *n*th number. The second integer *m* ($0 \le m \le 30$) indicates the number of required *1*s.

Your output should consist of the single number which solves the requested task. Input data will be chosen so that a valid solution exists and requires no more than 30 digits. Since your answer might contain as many as 30 digits, you should put a single space after every 6th digit in your answer (eg. output `110111 110` rather than `110111110`).

3 (b)
[7 marks]

Suppose we are interested in the *n*th number that has no more than *m* digits (where 0 is the 1st number). What is the answer when:

i. n=3 and m=4
ii. n =1 and m=24
iii. n=32 and m=5
iv. n=6410 and m=14
v. n=1000000 and m=21

3 (c)
[3 marks]

How many numbers are there between `1000001` and `11000001` (exclusive) using no more than 8 digits?

3 (d)
[5 marks]

What is the largest number of digits that can change between two consecutive numbers (in the new order), consisting of no more than *m* digits? Justify your answer.

Total marks: 100.

End of BIO 2003 Round One paper