British Informatics Olympiad

Time allowed: 3 hours

The 2017 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 an empty USB stick (or other storage device) on which to save your programs. You must not use any other material such as 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 *1 second* 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: *Coloured Triangles*

A *coloured triangle* is created from a row of squares, each of which is *red*, *green* or *blue*. Successive rows, each containing one fewer square than the last, are generated by considering the two touching squares in the previous row. If these squares are identical, the same colour is used in the new row. If they are different, the missing colour is used in the new row. This is continued until the final row, with only a single square, is generated.

In the following three example triangles `R`, `G` and `B` have been used to represent the colours. Note that each row is generated from the row above.

```
G G
 G
```

```
R B
 G
```

```
R R G B R G B B
 R B R G B R B
  G G B R G G
   G R G B G
    B B R R
     B G R
      R B
       G
```

1(a) [23 marks]

Write a program which reads in a starting row of between 1 and 10 (inclusive) uppercase letters (each `R`, `G` or `B`).

You should output the colour (`R`, `G` or `B`) of the square in the final row of the triangle.

Sample run 1

```
RG
B
```

Sample run 2

```
RBRGBRB
G
```

1(b) [3 marks]

How many possible rows of nine squares are there that generate `RRGBRGBB`? List them.

1(c) [4 marks]

Suppose you have a triangle where, on each row, only the colour of a single square is known. How many ways can the unknown squares be coloured so that the triangle is consistent with the rules for generating a *coloured triangle*? Justify your answer.

1(d) [3 marks]

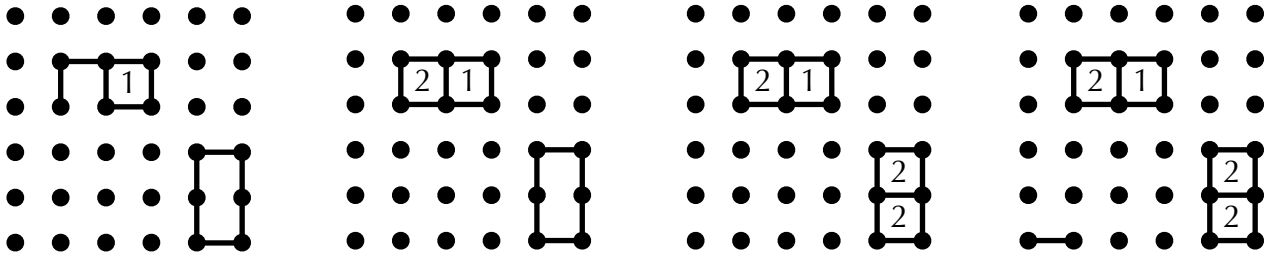
If the first row contains 4 squares, the colour of the square in the final row only depends on the extreme left and right of the first row. This is not true if, for example, the first row contains 5 squares. Find a larger size of row which shares this unusual property.

Question 2: *Dots and Boxes*

The game of *dots and boxes* is being played on a 6×6 grid of dots. Two players take turns selecting two adjacent dots (horizontally or vertically) and joining them by an edge. If the edge finishes off any (1×1) squares then those squares are won by the current player who then gets another turn, otherwise it becomes the other player's turn. Two adjacent dots can only be joined once.

For convenience in this question, we will number the dots 1 to 36, with the top row running from 1 (left) to 6 (right), the next row from 7 to 12 etc.

For example, the following sequence of grids shows a game in progress:



We start with the grid on the left where player 1 has won a single square. It is player 2's turn and they choose to join dots 14 and 15, finishing off the square towards the top left. Player 2 now gets another turn and chooses 29—30 which simultaneously finishes off both bottom right squares. Player 2 now gets another turn and joins 31—32 on the bottom left. It is now player 1's turn again.

In this question you will simulate two players who follow a very simple strategy.

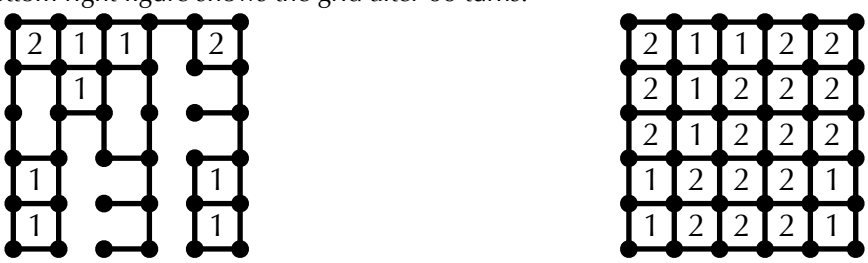
Each player uses a *modifier* and keeps track of a *position*. At the start of their turn, a player increases their position by their modifier and looks at the dot in the new position. If it is possible they then join an edge to that dot. If it is not possible they repeatedly increase the position by 1 (rather than the modifier) until they find a dot to which they can connect an edge.

A player increasing their position past dot 36 starts again at dot 1. E.g. If a position of 35 is modified by 3, the new position will be 2.

When trying to join an edge from a dot each player first looks to see if an edge can be added going upwards. If this is not possible, player 1 tries edges in a clockwise direction (i.e. first seeing if an edge goes to the right, then downwards and finally to the left) and player 2 tries edges in a counterclockwise direction.

For example, suppose no dots have been joined, player 1 starts with position 4 and a modifier of 10, and player 2 starts with position 14 and a modifier of 23:

- Player 1 increases their position to 14. An edge is added upwards joining 14—8.
- Player 2 increases their position to 1 (14 + 23 = 37, which is a single position beyond 36). An edge cannot be added upwards or to left (dot 1 is at the upper left corner), so an edge is added downwards joining dot 1 to dot 7.
- Player 1 increases their position to 24. An edge is added upwards joining 24—18.
- Player 2 increases their position to 24. An edge already exists upwards, so an edge is added to the left joining 24—23.
- The bottom left figure shows the grid after 46 turns; it is player 1's turn and they are at position 8. They look at position 18 (no possible edges), then 19 (no possible edges), then 20 and join 20—21.
- The bottom right figure shows the grid after 60 turns.



2(a) [24 marks]

Write a program that plays a game of *dots and boxes*.

Your program should first read five integers: the starting position p_1 ($1 \le p_1 \le 36$) then modifier m_1 ($1 \le m_1 \le 35$) for player 1, followed by starting position p_2 ($1 \le p_2 \le 36$) then modifier m_2 ($1 \le m_2 \le 35$) for player 2, followed by the number of turns to simulate t ($1 \le t \le 60$).

You should output a grid showing which squares have been won after t turns. Output an `X` for player 1, an `O` for player 2, and a `*` for a square that has not yet been won. This should be followed by the number of squares won by the first player, then the number of squares won by the second player.

Sample run

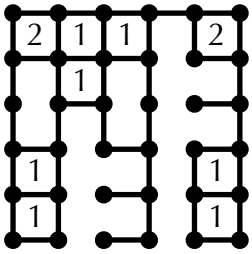
```
4 10 14 23 47

O X X * O
* X * * *
* X * * *
X * * * X
X * * * X

8 2
```

2(b) [2 marks]

Suppose the grid is in the following configuration and it is player 1's turn. If players can adopt any strategy (not just the very simple strategy), what is the maximum number of squares player 1 can win at the end of the game?



2(c) [3 marks]

How many different sets of input, where t=60, lead to player 2 winning all 25 squares?

2(d) [5 marks]

When an edge is added either 0, 1 or 2 squares are completed. Suppose the players are allowed to use any strategy, and the game is played until every edge has been added. If 2 squares have never been simultaneously completed during the game, what can you say about the pair of dots that were joined on the last turn? Justify your answer.

Question 3: *Mystery Parcel*

A shop is running a promotion and giving away *mystery parcels*, containing one or more items. As these are promotional, items are not very exciting (items that weigh the same are indistinguishable) and the total weight of items in each parcel is the same. Parcels containing the same combination of items (order does not matter) are themselves indistinguishable.

For advertising purposes the shop wishes to calculate the number of ways it can distribute the parcels.

For example, suppose there are 2 parcels containing a total of 4 items, each item weighing 1, 2 or 3 units. There are 10 different ways the parcels can be distributed. They might be constructed in 8 ways:

```
 1 1     —    1 1
 1 2     —    1 2
 2 2     —    2 2
 1 3     —    1 3
 2 3     —    2 3
 3 3     —    3 3
 3 1     —    2 2
1 1 1    —     3
```

In each of the first 6 pairings the parcels are indistinguishable, so there is only a single way in which they can be distributed. In each of the last 2 pairings the parcels can be distinguished, so they could each be distributed in 2 different ways; i.e. it can be distinguished which is distributed first.

3(a) [25 marks]

Write a program to determine the number of ways parcels can be distributed.

Your program should input four integers in order: p ($1 \le p \le 5$) indicating the number of parcels, i ($1 \le i \le 10$) indicating that items can weigh any integer from 1 to i inclusive, n ($1 \le n \le 25$) indicating the total number of items in all the parcels, and w ($1 \le w \le 25$) indicating the weight of each parcel.

You should output the number of ways parcels can be distributed. You will not be given input that requires an answer greater than 2^{31}.

Sample run

```
2 3 4 3
3
```

Marks are available for the case where $p = 1$

3(b) [2 marks]

How many ways can 3 parcels, containing a total of 6 items weighing 1, 2 or 3 units, be distributed?

3(c) [6 marks]

There are 92378 distinguishable parcels containing 10 items (when $i = 10$) and the shop has created an index of these parcels. Given two parcels, the one with the largest number of weight-10 items appears earliest in the index. If those are equal, the order depends on the number of weight-9 items, and so on. E.g. `8888855555` is before `8888855554` which is before `8888777777`.

What is the contents of the parcel at position 50,000? What is the position of the parcel containing one item of each weight from 1 to 10?

Total Marks: 100 End of BIO 2017 Round One paper