British Informatics Olympiad

Time allowed: 3 hours

Sponsored by Jane Street

The 2021 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.
- Some written questions can be solved by hand without solving the programming parts.
- The final written part of each question is intended to be a difficult challenge.
- Do not forget to indicate the name given to your programs on your answer sheet(s).

Question 1: *Down Pat*

A *pat* is a single letter or a string of letters which can be split into a left and right string (of at least 1 letter) where: each is the reverse of a *pat*; and all the letters in the left string are later in the alphabet than all the letters in the right string.

For example:

- `BA` is a pat as it splits into `B` and `A`, both of which are single letters and therefore pats, and `B` is alphabetically after `A`. `AB` is not a pat as the alphabetical rule would be broken;
- Similarly `ED` is a pat but `DE` is not;
- `DEC` is a pat as it splits into `DE` (whose reverse `ED` is a pat) and `C`.
- `CEDAB` splits into `CED` and `AB`, whose reverses are pats and `C`, `E`, and `D` are after `A` and `B` alphabetically.

1(a) [24 marks]

Write a program that reads in two strings from a line, s_1 then s_2, each between 1 and 6 *uppercase* letters inclusive.

You should output three lines, each containing a `YES` or `NO` indicating, in order, if s_1 is a pat, if s_2 is a pat, and if s_1s_2 (the combination of the two words) is a pat.

You must get all three lines of output correct to score marks.

Sample run

```
DE C
NO
YES
YES
```

1(b) [3 marks]

Which permutations of `ABCD` are pats?

1(c) [5 marks]

How many permutations of the alphabet, beginning with the letter `B`, are pats?

Question 2: *Tri-iso-game*

A game is being played on an infinite *triangular grid*, made up from *equilateral triangles* (with side length of 1 unit). They are regularly tiled, with each edge (except at the ends) of a triangle touching one edge of another triangle. The grid is orientated so that each triangle has one edge horizontal.

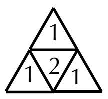
At the start of the game a single point-upwards triangle contains a `0`; all other triangles are *empty*. On each move of the game an empty triangle, which is adjacent to a triangle already containing a number, will be filled with a number. Player 1 will fill triangles with `1`s, player 2 will fill triangles with `2`s, etc. Players take it in turns to move; with *n* players this would be 1, 2, …, *n*, 1, 2, …, *n*, 1, …

When deciding which triangle to fill, players will traverse around the perimeter of the filled triangles. Each player is positioned on one triangle's edge (which is on this perimeter) and will traverse by moving to an adjacent edge. All players move around the perimeter in the same direction for the entire game. Note that players are positioned on edges and *not* in the triangles adjacent to edges.

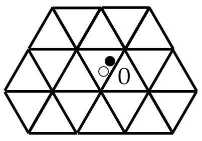
Players start against the left-hand edge of the triangle containing a `0`. After a single traversal they will move to the right-hand edge; i.e. clockwise around the triangle.

Each player has a maximum number of traversals they can use each move. After they have finished traversing the perimeter, they fill the empty triangle, adjacent to where they ***started*** the move, with their number. If, after a triangle is filled, *any* player finds that their current position is no longer touching an empty triangle, they reposition themselves on the left-hand edge of the left-most filled triangle on the highest row of the grid.

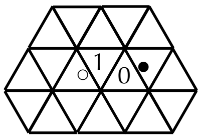
Four adjacent triangles (of length 1) can be seen as larger triangle (of length 2). A player scores 1 point for all such larger triangles where the three outer triangles are filled with their number. It does not matter if the inner triangle is empty or contains a different number. For example, the figure to the right is worth 1 point to player 1.



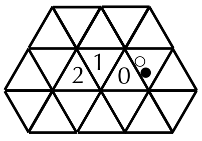
Players try to set themselves up for future turns. When traversing the perimeter, a player will stop before their maximum number of traversals in the following situation: they have reached an edge (after at least one traversal) whose empty adjacent triangle would score them a point *if* it were filled in. Note that it is not actually filled when they stop.



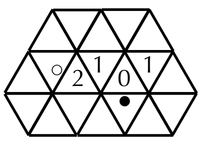
For example, suppose the game has 2 players. Player 1 can traverse sixteen edges a move; player 2 can traverse two edges. In the diagrams to the right, ● and ○ show the positions of the two players.



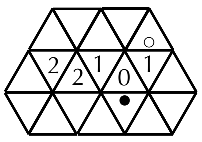
- Initially, there is a single filled triangle containing a `0` and both player 1 (●) and player 2 (○) are positioned against its left edge;
- Player 1 traverses sixteen edges, looping around the `0` triangle until finishing against its right edge. They then fill in the triangle adjacent to where they started with a `1`. As player 2 is no longer adjacent to an empty triangle they are repositioned.



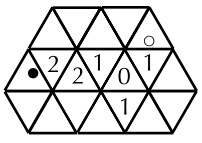
- Player 2 now traverses two edges, finishing in the same position as player 1. They fill the triangle adjacent to their starting edge with a `2`.
- Player 1 starts to traverse the perimeter. Note that they do not stop after the first edge has been traversed; if they filled in the triangle at this stage they would not gain a point as the triangle adjacent to their starting position is not placed until the move's end. After sixteen edges they eventually stop and fill in a triangle. Once again Player 2 is repositioned.



- Player 2 moves two edges. Note that, even though two successive edges were horizontal, it still takes two traversals. A triangle is filled in.



- Player 1 moves and stops after 7 traversals as filling in the adjacent triangle would score them a point. They do not fill in the triangle on stopping. Their traversals now being completed they fill in their starting triangle, which is co-incidentally where they had stopped, score a point and are then repositioned.



2(a) [24 marks]

Write a program that plays the game.

Your program should first input a line containing two integers, p ($1 \leq p \leq 5$) then m ($1 \leq m \leq 5,000$), indicating the number of players and a number of moves to play. This will be followed by a line containing p integers, each between 1 and 100 inclusive, the i^{th} indicating the maximum number of traversals player i can make each move.

You should first output p lines. The i^{th} line should contain the final score for player i. This should be followed by a line containing the length of the *outside* perimeter of the filled triangles at the end of the game.

Sample run

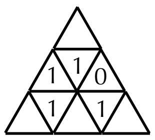
```
2 5
16 2
1
0
8
```

2(b) [4 marks]

In a 1 player game, what does the grid look like after 5 moves if the player can make 1 traversal on each move? What does the grid look like after 5 moves if the player can make 2 traversals each move?

2(c) [3 marks]

In a 1 player game, after 4 moves the grid is filled as in the diagram. What is the smallest possible value for the player's maximum number of traversals each move?



2(d) [5 marks]

In a 2 player game, where player 1 can make 5 traversals per move and player 2 can make 7 traversals per move, how many unfilled triangles are inside the perimeter after 5000 moves?

Question 3: *Window Dressing*

A shop is looking to display some boxes (conveniently labelled `A`, `B`, …) in their window. There is a desired order for the boxes to appear in the window but unfortunately the boxes arrive from the warehouse in alphabetical order and access to the window limits how the boxes can be added to the display.

There are three possible operations the store manager can employ to change the order of the boxes in the window:

- (Add) The next box from the warehouse can be added to the end of the boxes currently on display;
- (Swap) The first two boxes in the window display can be swapped;
- (Rotate) The first box can be removed from the display and replaced at the other end of the display.

Given a desired order for the display, the manager wishes to find the minimum number of operations needed.

For example, suppose the desired order is ACBD:

- The manager could add A then B then C then D (giving ABCD after 4 operations), then rotate (giving BCDA), swap (CBDA) and rotate another 3 times (ACBD). A total of 9 operations.
- The manager could add A then B (giving AB), then swap (BA), add C (BAC), rotate (ACB) and add D (ACBD). A total of 6 operations.

3(a) [24 marks]

Write a program to determine the minimum number of operations to display the boxes.

Your program should input a string of *b* ($1 \leq b \leq 8$) uppercase letters, a permutation of the first *b* letters of the alphabet.

You should output the minimum number of operations required to reach this permutation.

Sample run

```
ACBD
6
```

3(b) [3 marks]

Which orderings of the 5 boxes `A`, …, `E` require a minimum of 6 operations?

3(c) [5 marks]

The arrangement HGFEDCBA takes a minimum of 24 operations. How many different sequences of 24 operations lead to this arrangement?

Total Marks: 100

End of BIO 2021 Round One paper