British Informatics Olympiad

Time allowed: 3 hours

The 2020 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: *Roman Look-and-Say*

The *Roman look-and-say* description of a string (of `I`s, `V`s, `X`s, `L`s, `C`s, `D`s and `M`s) is made by taking each block of adjacent identical letters and replacing it with the number of occurrences of that letter, given in *Roman numerals* **(*)**, followed by the letter itself. A block of adjacent identical letters is never broken into smaller pieces before describing it.

For example:

- `MMXX` is described as "two `M`s followed by two `X`s". Since two is `II` in Roman numerals, this is written as `IIMIIX`;
- `IIMIIX` is described as `IIIIMIIIIX`, which is "two `I`s, one `M`, two `I`s, one `X`";
- `IIIIMIIIIX` is described as `IVIIMIVIIX`;
- It is *not* valid to describe `III` as, "two `I`s, one `I`" `IIIII`.

Note that Roman look-and-say descriptions are *not* necessarily Roman numerals.

1(a) [25 marks]

Write a program that reads in a Roman numeral representing a number between 1 and 3999 inclusive, followed by n ($1 \leq n \leq 50$).

You should apply the *Roman look-and-say* description n times and then output the number of `I`s in the final description, followed by the number of `V`s.

Sample run 1

```
MMXX 1
4 0
```

Sample run 2

```
MMXX 3
6 2
```

1(b) [2 marks]

How many Roman numerals (from 1 to 3999 inclusive) have a Roman look-and-say description that is also a Roman numeral? List these *descriptions*.

1(c) [4 marks]

The Roman look-and-say descriptions are generated for all the Roman numerals from 1 to 3999 (inclusive). How many distinct descriptions are there?

(*) *Roman numerals* are conventionally defined to represent numbers using seven letters: `I`=1, `V`=5, `X`=10, `L`=50, `C`=100, `D`=500 and `M`=1000. Numbers other than these are formed by placing letters together, from left to right, in descending order of size, and adding their values. The basic rule is to always use the biggest numeral possible (e.g. 15 is represented as `XV` but never as `VVV`, `VX` or `XIIIII`).

Letters may not appear more than three times in a row, so there are six exceptions to these rules – the combinations `IV`, `IX`, `XL`, `XC`, `CD` and `CM`. In these cases a letter is placed before one of greater value and the smaller value is subtracted from the larger. E.g. `CD` = 400. These are the *only* exceptions so, for example, `MIM` is not valid.

Question 2: *Alpha Complex*

Alpha Complex consists of many connected rooms. Each room is identified by a single (unique) letter and each exit from a room is marked with the letter identifying the room to which it is connected.

A *spy* is moving systematically through Alpha Complex. For each room they keep track of whether they have visited it an even or an odd number of times. They also record, for each room and for each exit, whether they have left the room through that exit an odd or even number of times.

The spy moves between rooms according to the following rules:

- If they have visited the room an odd number of times, they will leave through the exit which is marked with the first letter alphabetically;
- If they have visited the room an even number of times, they will find the first exit alphabetically that they have left through an odd number of times. If that is the last exit alphabetically in this room they will leave through it, otherwise they will leave through the next exit alphabetically.

When the spy starts exploring Alpha Complex, they have visited their starting room an odd number of times (i.e. once), each other room an even number of times (i.e. zero) and have left through each exit an even number of times (i.e. zero).

For example, suppose that they start in room `X`, which has exits to `B`, `I` and `O`:

- They have visited `X` an odd number of times, so they leave the room through the first exit alphabetically, which is `B`;
- When they are next in X they will have visited it an even number of times. The first exit alphabetically that they have left through an odd number of times is `B`. As this is not the last exit alphabetically, they leave through exit `I`;
- Subsequent visits to the room will see them leave through `B`, `O`, `B`, `I`, `B`, `O`, …

Alpha Complex consists of r ($r \geq 3$) rooms, identified by the first r letters of the alphabet. The spy is in possession of a secret *plan*, giving the connections between the rooms. This plan is an ordered list of $r-2$ letters and the spy can construct a map of the complex as follows:

- The spy will *choose* the first room alphabetically which has not yet been chosen and which is not in the plan. The chosen room is connected to the first room in the plan. The first room is then removed from the plan;
- The above step is repeated until the plan is empty;
- There will be two rooms which have not yet been chosen. These two rooms are connected together.

If two rooms are connected, each room has an exit to the other room.

For example, suppose Alpha Complex has 6 rooms and the plan is `E`, `A`, `E`, `D`:

- The first room which is not in the plan is `B`, which gets connected to `E` (the first room in the plan). The updated plan becomes `A`, `E`, `D`. Note that only `B` has been chosen and that the other instance of `E` in the plan is not removed;
- The next chosen room will be `C`, which gets connected to `A`;
- The next chosen room will be `A` (it has not been previously chosen and is no longer in the plan) and it is connected to `E`;
- `E` is now chosen and connected to `D`;
- The plan is now empty. As `D` and `F` were never chosen, they are now connected.

2(a) [24 marks]

Write a program that tracks the spy through Alpha Complex.

Your program should first input a string consisting of n ($1 \le n \le 8$) uppercase letters (each from the first n+2 letters of the alphabet) indicating the spy's plan of Alpha Complex, followed by p and q ($1 \le p < q \le 1{,}000{,}000$) indicating moves the spy will make.

The spy starts in room `A`.

You should first output n+2 lines. The first containing (in alphabetical order) the rooms connected to `A`, the second those connected to `B`, etc. These should be followed by a line containing the spy's location after p and q moves.

Sample run

```
EAED 1 2
CE
E
A
EF
ABD
D
CA
```

2(b) [2 marks]

What is the plan if Alpha Complex has 3 rooms and the connected rooms are `A-B` and `A-C`? What is the plan if there are 6 rooms and the connections are `A-B`, `A-C`, `A-D`, `A-E` and `A-F`?

2(c) [4 marks]

After an unknown number of moves in Alpha Complex the spy arrives in a room having forgotten if it has been visited an odd or an even number of times. How can they continue their systematic exploration without deviating from their original intended route? Justify your answer.

2(d) [4 marks]

If Alpha Complex contains 8 rooms, how many different plans are there where the first four connections the spy adds when constructing the map are (in some order) `A-E`, `B-F`, `C-G` and `D-H`?

Question 3: *False Plan*

A spy, currently working their way through an enemy compound, has been given a *false plan*. The plan is an ordered list of letters. In order to make the plan look realistic, the number of adjacent identical letters in the plan has been limited.

For example, suppose that the plan contains four letters, each of which is an `A` or a `B`, and that there are never more than two adjacent identical letters. There are 10 possible plans:

```
AABA
AABB
ABAA
ABAB
ABBA
BAAB
BABA
BABB
BBAA
BBAB
```

These have been listed in *alphabetical* order.

3(a) [27 marks]

Write a program to determine the n^{th} possible plan.

Your program should input a line containing three integers p, q, r ($1 \le p, q, r \le 12$) indicating (in order) that the first p letters of the alphabet can be used, no more than q adjacent identical letters are permitted and that the plan should contain exactly r letters. You should then input a line containing a single number n ($1 \le n < 2^{63}$).

You will only be given input where n is no greater than the number of possible plans.

You should output the n^{th} possible plan.

Sample run

```
2 2 4
7
BABA
```

3(b) [3 marks]

Suppose `A`s, `B`s, `C`s and `D`s are permitted and there are never more than two adjacent identical letters. Consider the list of possible three letter plans; which plan is `CCA` within this list? How about (for eight letter plans) `CCABABCC`?

3(c) [5 marks]

Suppose there is a plan that is in the n^{th} position both when the plans are ordered alphabetically and when they are ordered reverse alphabetically. What can you determine about p, q, and r? Justify your answer.

Total Marks: 100 End of BIO 2020 Round One paper