British Informatics Olympiad

Time allowed: 3 hours

The 2015 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: *Block Palindromes*

A *palindrome* is a word that shows the same sequence of letters when reversed. If a word can have its letters grouped together in two or more blocks (each containing one or more adjacent letters) then it is a *block palindrome* if reversing the order of those blocks results in the same sequence of blocks.

For example, using brackets to indicate blocks, the following are *block palindromes*:

- BONBON can be grouped together as (BON)(BON);
- ONION can be grouped together as (ON)(I)(ON);
- BBACBB can be grouped together as (B)(BACB)(B) or (BB)(AC)(BB) or (B)(B)(AC)(B)(B)
  Note that (BB)(AC)(B)(B) is *not* valid as the reverse (B)(B)(AC)(BB) shows the blocks in a different order.

1(a) [23 marks]

Write a program which reads in a word of between 2 and 10 (inclusive) uppercase letters.

You should output a single number, the number of different ways the input can be grouped to show it is a *block palindrome*.

Sample run

```
BBACBB
3
```

1(b) [2 marks]

Give all the groupings of AABCBAA that show it is a *block palindrome*.

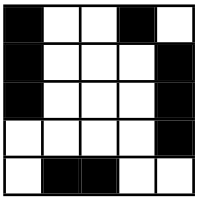
1(c) [6 marks]

Suppose that all the groupings of a *block palindrome* contain an even number of blocks. What can you say about the length of the *block palindrome*? How many different groupings can it have? Justify both your answers.

Question 2: *Battleships*

In the game of *battleships* players secretly position ships of various sizes on a board and then try to determine the position of their opponent's ships. In this question we will consider the placement of one player's ships on a 10×10 board of squares. Each ship is placed either horizontally or vertically, covering an exact number of adjacent squares which are all on the board; a ship of size 1×*n* is called an *n*-ship. No part of any ship can be placed in a square that is adjacent (horizontally, vertically or diagonally) to part of another ship.

For example, part of a board showing four ships (two *3*-ships, a *2*-ship and a *1*-ship) is shown to the right. This is ***not*** valid as the *1*-ship is touching one of the *3*-ships diagonally. If the 1-ship was moved one square to the left everything shown would be valid.



The bottom left corner of the board has co-ordinate (0, 0). Co-ordinates are given as (x,y).

Our player starts by choosing (non-negative integer) values for `a`, `c`, `m` and `r` and then places each ship in turn using the following algorithm:

We are using the notation X←Y to mean "set X to Y"
X *mod* Y is equivalent to the remainder when X is divided by Y

1. `r` ← (`a` × `r` + `c`) *mod* `m`
2. Use the units digit of `r` as an *x* co-ordinate and the tens digit of `r` as a *y* co-ordinate
3. `r` ← (`a` × `r` + `c`) *mod* `m`
4. Starting at the calculated co-ordinates and *only* if it is valid, if `r` is even place the ship going horizontally to the right or if `r` is odd place it vertically upwards
5. If the ship is placed stop, otherwise continue from step 1.

Step 3 is never skipped, even if the value of `r` does not affect the validity of the ship in step 4.

For example, suppose the board contains a single *1*-ship at co-ordinate (3,0), that `a`, `c`, `m` and `r` are currently 3, 5, 53 and 20 respectively, and that the player is trying to place a *2*-ship on the board.

First `r` is set to 12 (3×20+5 = 65 and 65 *mod* 53 = 12) which represents co-ordinate (2,1), then `r` is set to 41 indicating that the ship is placed vertically upwards from (2,1) if valid. This is *not* valid as the ship would occupy (2,1) and (2,2) and the first square is diagonally adjacent to (3,0). `r` is then set to 22 representing co-ordinates (2,2), then set to 18 indicating that the ship is placed horizontally in squares (2,2) and (3,2) which is valid.

It is possible, for some values, that this algorithm will never find a valid position for the ship.

2(a) [27 marks]

Write a program that places ships in a game of battleships.

Your program should first read in three integers: a ($1 \leq a \leq 2^{15}$) then c ($1 \leq c \leq 2^{15}$) and finally m ($1 \leq m \leq 2^{15}$); the initial value of r (which is not input) is always 0. You should then follow the player's algorithm to place a *4*-ship, two *3*-ships, three *2*-ships and four *1*-ships (in that order) on an initially empty board.

You will only be given input that allows all the ships to be placed on the board.

You should output ten lines, the i^{th} of which containing information for the i^{th} placed ship. Each line should contain the x then y starting co-ordinates for its ship, followed by an `H` or a `V` indicating whether the ship is placed horizontally or vertically. (For a *1*-ship you should still indicate the appropriate `H` or `V` based on the r value.)

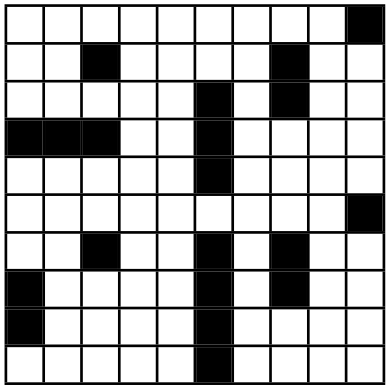
Sample run

```
10 5 9999
```

```
5 0 V
5 5 V
0 6 H
0 1 V
7 2 V
7 7 V
2 8 H
2 3 V
9 4 V
9 9 H
```

2(b) [2 marks]

If a, c, m and r are set to 2, 3, 17 and 0 respectively, only four co-ordinates will ever be produced by the algorithm in step 2. What are they?



2(c) [3 marks]

The grid to the right corresponds to the sample run.

Another player secretly moves one ship to a valid position. A square can be guaranteed as empty if it is not possible for any ship to occupy it after this move. How many such squares are there?

2(d) [5 marks]

How many valid arrangements are there, on a 5x5 board, of a *4*-ship, a *3*-ship a *2*-ship and a *1*-ship? (All four ships must be on the board.)

Question 3: *Modern Art*

A gallery is displaying pieces of *modern art* by several artists and is considering the different ways of arranging the exhibition. As this is modern art all pieces by the same artist are indistinguishable.

For example, suppose there is a single piece by artist `A`, two by artist `B` and one by artist `C`. There are 12 ways the gallery might arrange the exhibition:

```
ABBC
ABCB
ACBB
BABC
BACB
BBAC
BBCA
BCAB
BCBA
CABB
CBAB
CBBA
```

These have been listed in *alphabetical* order.

3(a) [25 marks]

Write a program to determine the n^{th} way of arranging the exhibition.

Your program should input five integers: *a*, *b*, *c* and *d* (each between 0 and 5 inclusive) indicating the number of works by artists `A`, `B`, `C` and `D` in order, and finally *n* ($1 \leq n \leq 2^{34}$).

You will only be given input where at least one artist is exhibiting a work and *n* is no greater than the number of possible exhibitions.

You should output the string which represents the n^{th} arrangement.

Sample run

```
1 2 1 0 8
BCAB
```

3(b) [2 marks]

If the gallery is exhibiting `AABCCBDD` which arrangement is this?

3(c) [5 marks]

An unspecified number of artists are exhibiting an unspecified number of pieces of modern art. The gallery has exhibited the n^{th} arrangement followed by the $n+1^{st}$ arrangement. For poetic reasons, for each position in the n^{th} arrangement the artist providing the work in the same position in the $n+1^{st}$ arrangement was different. Is it possible that, in the $n+2^{nd}$ arrangement, the artist providing the work in each position is different to that in the $n+1^{st}$ arrangement? Justify your answer.

Total Marks: 100 End of BIO 2015 Round One paper