British Informatics Olympiad

Time allowed: 3 hours

Sponsored by Jane Street

The 2026 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, code written outside of the contest, AI generated code, 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: Grid Notation

Consider a 5×5 *grid* whose squares are labelled by letters of the alphabet as follows:

```
A B C D E
F G H I J
K L M N O
P Q R S T
U V W X Y
```

For larger grids we can repeatedly sub-divide them into 5×5 sections. We will only consider grids of size $5^n \times 5^n$.

The labelling for a section is the sequence of labels as the grid is divided, with labels for the larger sections appearing to the left. Labels can be matched to (x, y) co-ordinates of squares in the grid; the bottom-left square will be (1, 1).

For example:

- A 125×125 grid would be divided into 5×5 sections (each of size 25×25), and those would be further divided into 5×5 sub-sections (each of size 5×5);
- In the first division, co-ordinates (1, 1) and (25, 25) are the corners of the section whose label is `U`. Co-ordinates (26, 101) and (50, 125) are the corners of section `B`;
- In the second division, co-ordinates (41, 116) and (45, 120) are the corners of a subsection of `B`, whose label is `BI`;
- Co-ordinate (45, 118) is a square in subsection `BI`, whose label is `BIO`.

1(a) [22 marks]

Write a program that reads in a label of ($1 \leq n \leq 10$) uppercase letters (excluding `Z`s).

You should output the co-ordinates of the square, in a $5^n \times 5^n$ grid, corresponding to the input label.

Sample run

```
BIO
45 118
```

1(b) [3 marks]

What is the label of co-ordinate (209, 217) in a 625×625 grid? What is the label of (606445, 9161058) in a 9765625×9765625 grid?

1(c) [5 marks]

In a 3125×3125 grid, how many squares are directly adjacent (horizontally or vertically) to a square with a label that shares no letters?

Question 2: Playing Codes

Two *decks* of playing cards (*alpha* and *beta*), whose faces show individual letters rather than the usual suits and values, are being used to encrypt (and decrypt) words. Each deck contains n (≥ 5) cards, one for each of the first n letters of the alphabet.

A letter in the word is encrypted by *simultaneously* taking the top card from each deck and placing it at the bottom of the same deck, then repeating this operation until the top card on the alpha deck shows the letter we wish to encrypt. The letter then showing on top of the beta deck is the encrypted letter. We will only ever try to encrypt letters that appear in the alpha deck.

When showing the order of a deck we will show the top card on the left.

For example, suppose the alpha deck is in the order `EAFBCD` and the beta deck is in the order `AEFBDC`:

- If we wanted to encrypt `E` the cards would already be in position, and the encrypted letter would be `A`;
- If we wanted to encrypt `A` the decks would be changed in a single operation to `AFBCDE` and `EFBDCA`. The encrypted letter would be `E`;
- If we wanted to encrypt `B`, after multiple operations the decks would be in the order `BCDEAF` and `BDCAEF`. The encrypted letter would be `B`.

After encrypting a letter, the decks are further manipulated before the next letter is encrypted. The *second* card from the alpha deck is removed and placed into the middle position of that deck. For the beta deck, the top card is first placed on the bottom and then the *third* card in the deck is placed in the middle position. If the deck contains an even number of cards, the *middle* is selected so the number of cards above it is 1 more than the number below.

For example, suppose after encryption the decks are (alpha) `BCDEAF` and (beta) `BDCAEF`:

- The alpha deck becomes `BDECAF` after `C` is moved to the middle;
- The beta deck first moves `B` to become `DCAEFB` and then `A` is moved giving `DCEAFB`.

The letters in a word will be encrypted in order (from left to right).

Before encryption begins the starting order of the decks will be generated as follows:

1. The combined cards from the two decks are placed together, so that they are in alphabetical order; i.e. `AABBCCDD`… Cards with the same letter are indistinguishable;
2. The combined deck is split at the mid-point into two halves;
3. Cards are alternately taken from the top of the two halves, starting with the original lowest half, and a new combined deck formed by adding those cards, one at a time, to the bottom of the new deck;
4. The top card of the new deck is placed on the bottom of the deck, and this operation is repeated c times;
5. Steps 2–4 (splitting a combined deck, creating a new combined deck and c manipulations of the top card) are repeated s times;
6. This deck is now split into the alpha and beta decks so that first occurrence of each letter in the combined deck goes to the alpha deck, and the second occurrence goes to the beta deck. The order is otherwise maintained.

For example, suppose n is 6, c is 2 and s is 3:

- Starting with a deck `AABBCCDDEEFF`;
- The deck is split to `AABBCC` and `DDEEFF`, and those are combined to become `DADAEBEBFCFC`;
- Two cards are moved to the bottom giving `DAEBEBFCFCDA`;
- This is repeated again: splitting the deck to `DAEBEB` and `FCFCDA`; combining to `FDCAFECBDEAB`; adjusted to `CAFECBDEABFD`;
- This is repeated again: splitting the deck to `CAFECB DEABFD`; combining to `DCEAAFBEFCDB`; adjusted to `EAAFBEFCDBDC`;
- The first occurrence of each letter, without otherwise adjusting the order, gives an alpha deck of `EAFBCD`. The beta deck is `AEFBDC`.

2(a) [25 marks]

Write a program that encrypts words.

Your program should input a line containing three integers, n ($5 \leq n \leq 26$) then s ($0 \leq s \leq 5000$) then c ($0 \leq c \leq 51$), followed by a string of at most 10 uppercase letters (from the first n letters of the alphabet).

You should output a string giving the encrypted input string.

Sample run

```
6 3 2 BB
BD
```

2(b) [2 marks]

Suppose alpha and beta are both initially `ABCDEFGHIJKLMNOPQRSTUVWXYZ` and the resulting encrypted word is `LPWGZZ`. What was the original string?

2(c) [4 marks]

Suppose alpha is `ABCDEFGHIJKLMNOPQ`, and the word `ABCDEFGHIJKLMNOPQ` is encrypted. Give a beta which produces the encrypted word which is last in alphabetical order (compared to encrypted words from other betas).

2(d) [5 marks]

Suppose alpha is `ABCDEFGH`. How many betas are there, with `A` on top, where a 10-letter word (beginning with an `A`) can be found that encrypts to itself?

Question 3: Raindrop Numbers

A *raindrop* is an integer which contains exactly one digit smaller than the digit immediately to its left. In other words, running from left to right, there is only a single drop between adjacent digits.

For example:

- 45561 is a raindrop number as only digit 1 has a larger digit immediately to its left;
- 456 is not a raindrop number as no digits have a larger digit immediately to their left;
- 987 is not a raindrop number as both 8 and 7 have larger digits immediate to their left.

3(a) [24 marks]

Your program should input a single number n ($1 \leq n < 2^{63}$).

You should output a single integer giving the number of raindrops between 1 and n (inclusive).

Sample run

```
99
45
```

3(b) [2 marks]

What is the first raindrop greater than 8102026?

3(c) [4 marks]

If raindrops are listed in increasing numerical order, what is the 10,000,000,000th?

3(d) [4 marks]

Consider the list of raindrops, of ≤ 100 digits, in increasing numerical order. We can look at pairs of raindrops, of identical length, that are directly adjacent in the list. For any such pair, we can then determine the number of positions where the digits differ. E.g. 54 and 60 are successive raindrops and differ in both the first and second position; 5≠6 and 4≠0.

How many pairs of adjacent identical length raindrops are there which differ in all but one position?

Total Marks: 100

End of BIO 2026 Round One paper