British Informatics Olympiad

Time allowed: 3 hours

The 2018 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: *Debt Repayment*

At the end of every month interest is added to a *debt* (initially 100) and then repayments are taken off. The *interest* is a fixed percentage of the current debt. The *repayment* is also a fixed percentage of the debt (after the interest has been added) or 50, whichever is larger. If the repayment is greater than the debt it is reduced to match the debt. The cycle of interest and repayment is continued until the debt has been reduced to 0 and paid off.

For example, suppose the interest is 10% and repayments are 50%:

- At the end of the first month the interest is 10, increasing the debt to 110;
- The repayment is 55 (50% of 110) leaving the debt at 55;
- At the end of the second month the interest is 5.5, increasing the debt to 60.5;
- The repayment is 50 (as 50% of 60.5 is less than the minimum amount), leaving the debt at 10.5;
- At the end of the third month the interest is 1.05, increasing the debt to 11.55;
- The repayment is 11.55 (the minimum payment of 50 is reduced to match the debt) and the debt is paid off.
- The total amount repaid on the debt is 116.55

When calculating percentages the amount is rounded *up* to 2 decimal places. E.g. 10.201 and 10.207 would both be rounded up to 10.21.

1(a) [26 marks]

Write a program that reads in the interest percentage followed by the repayment percentage. Percentages will be integers between 0 and 100 (inclusive).

You should output the total amount repaid.

You will always be given input that allows the debt to be paid off.

Sample run 1

```
10 50
116.55
```

1(b) [2 marks]

Given an interest percentage of 43% and a repayment percentage of 46%, how many payments will be made to pay off the debt?

1(c) [3 marks]

Which interest and repayment percentages (integers between 0 and 100) lead to the largest amount repaid on a debt that is paid off?

Question 2: *Decoder Ring*

A *decoder ring* consists of two adjacent dials and is used to encrypt (or decrypt) messages. The two dials are lined up, so that each position on the first is touching one on the second, and dials can rotate so that they can be aligned in different ways. The first dial has 26 positions, lettered from A to Z in order, and the second dial has the same letters but not necessarily in the same order. A letter is encrypted by finding it on the first dial and using the corresponding touching letter on the second dial.

For example, suppose that the second dial has been lettered from Z to A (i.e. reverse order) and that the A on the first dial is touching Z on the second:

- The letter A would be encrypted to the letter Z, the letter B by the letter Y etc.
- If the second dial is now rotated until the A on the first dial is touching X on the second, the letter A would be encrypted to the letter X, the letter B by W etc.

The order of the letters on the second dial will be generated as follows, from the number n:

- Place the letters A to Z clockwise in order around a circle;
- Starting with A count clockwise round the circle until n is reached;
- *Remove* the selected letter from the circle — it becomes the first letter on the second dial;
- Starting from where you left off, count to n again to select the next letter;
- Continue until all the letters have been selected.

For example, if n is 5 the letters will be chosen in the order: E, J, O, T, Y, D, K, Q, W, ...

The dials will be aligned so that the first letter selected for the second dial is initially touching the letter A on the first dial, the second letter selected touching B etc. After each rotation of the dials, the letter on the second dial previously touching B on the first dial will be touching A on the first dial, etc.

A word is encoded by encrypting each letter in turn, after each encryption rotating the dial by a single position.

For example, if n is 5 the word ABCD will be encrypted as EOYK.

2(a) [24 marks]

Write a program that encrypts a word.

Your program should first input a single integer n ($1 \leq n \leq 1000000$) followed by a word w of between 1 and 8 upper case letters.

You should output the first 6 letters of the second dial generated from n, followed by the encrypted version of w.

Sample run

```
5 ABCD
EJOTYD
EOYK
```

2(b) [3 marks]

What are the first 6 letters of the second dial generated from n = 1,000,000,000?

2(c) [4 marks]

The word ABCDEFGHIJKLMNOPQRSTUVWXYZ is to be encrypted. Does a second dial exist that will encrypt this to a word that contains all 26 letters? Justify your answer.

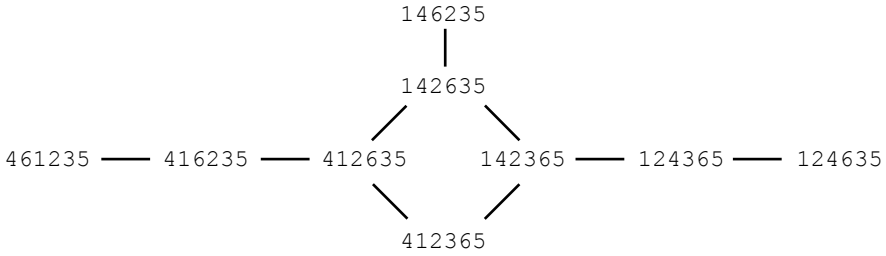
2(d) [4 marks]

A word is encrypted *without rotating the dials* and this encrypted word encrypted again and again until the first time the original word is produced. This will always occur eventually. What is the largest number of encryptions that might be required?

Question 3: *Serial Numbers*

A *serial number* can be transformed into another, by swapping two adjacent digits, so long as one of those two digits is adjacent to a digit whose value lies between them. Two *serial numbers* will be called *equivalent* if there is a sequence of transformations which changes one into the other, and the *distance* between them is the *minimum* number of such transformations required.

For example, the following shows all the serial numbers that are equivalent to `146235`, the lines showing all the valid transformations:



The distance from `461235` to `124635` is 6, and from `412635` to `142635` is 1.

3(a) [24 marks]

Write a program that determines the largest distance between a *serial number* and any equivalent number.

Your program should input a single integer d ($1 \leq d \leq 9$) indicating the number of digits in the serial number, followed by a d digit serial number which will consist of the numbers 1,...,d without repetition.

You should output a single integer giving the largest distance.

Sample run

```
6
461235
6
```

3(b) [4 marks]

What is the largest distance between *any* two serial numbers equivalent to 326451? How about 183654792?

3(c) [6 marks]

What is the size of the largest set of serial numbers, none of which are equivalent, each consisting of the digits 1,...,5 without repetition? What is the size if the serial numbers consist of the digits 1,...,9 without repetition?

Total Marks: 100

End of BIO 2018 Round One paper