British Informatics Olympiad

Time allowed: 3 hours

The 2008 British Informatics Olympiad

LIONHEAD STUDIOS

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 a blank floppy disk (or other storage device) on which to save your programs. You must not use any other material such as disks, 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. The output format of your programs should follow the 'sample run' examples. Your programs should take less than 2 seconds 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: ***Goldbach Conjecture***

A *prime number* is a whole number, greater than *1*, that can only be divided by itself and the number *1*. It is known that all even numbers between *4* and *300,000,000,000,000,000* are equal to the sum of two primes (a fact that is believed to be true for all larger even numbers as well, and called the *Goldbach Conjecture*).

For example, *30 = 7 + 23*. There are two other ways of expressing *30* as the sum of two primes, which are *11 + 19* and *13 + 17*. These are the only ways of expressing *30* as the sum of two primes, since the order of the numbers in the additions does not matter.

1(a) [25 marks]

Write a program which inputs a single *even* number (between *4* and *10,000* inclusive) and outputs a single number which is the number of different ways the input can be expressed as the sum of two primes.

Sample run

```
22
3
```

1(b) [3 marks]

There are four ways of expressing *46* as the sum of two primes. What are they?

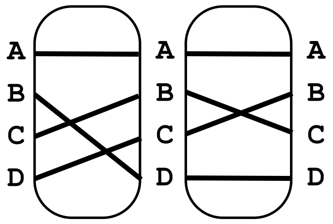
1(c) [2 marks]

There are many odd numbers which cannot be expressed as the sum of two primes. How many such numbers are there between *4* and *50*?

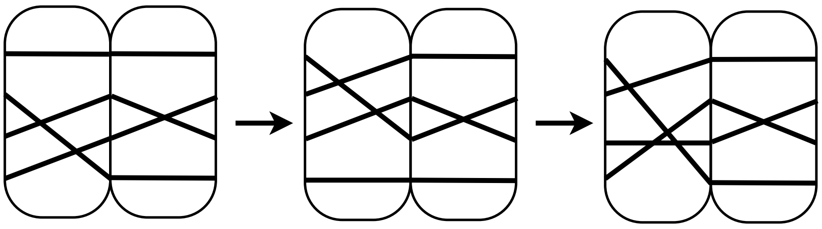
Question 2: *Enigma Machine*

The *Enigma machine* was a device used to encrypt (and decrypt) text during the second world war. Your task in this question is to simulate a simplified version of this machine. Our simulation, which will encrypt the letters A, B, C and D, has two components: rotors and reflectors.

A rotor has a left side and a right side, each of which has four ports (A, B, C and D). Internal wiring links ports, so that each port on the left is linked to exactly one port on the right (and hence each port on the right will be connected to exactly one port on the left). Our simulation has two rotors which are placed next to each other so that the ports with the same value touch. Their wiring and starting positions are:

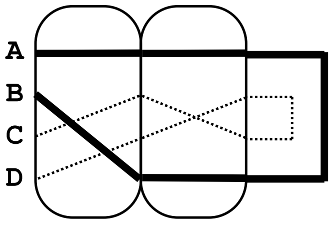


These rotors can turn (in the original Enigma machine they were disks). When a rotor turns, the ports (on both sides) which previously had the value D change to the value C, those which were C become B, Bs become As and As become Ds. The rotors are still arranged so that ports with the same values touch.


first rotor turning twice

On the right of the rotors is a reflector, which does not turn, and only has ports on its left side. Port A is wired to port D, and port B is wired to port C. The reflector is placed next to the second rotor so that the ports with the same value touch.

To encrypt a letter, we find its port on the left of the first rotor, and then follow the wiring through both rotors and the reflector until we get back to the left of the first rotor. The port we finish at gives us the encrypted version of our letter. After each letter has been encrypted we turn the first rotor. After every fourth letter has been encrypted we turn the second rotor; this happens ***in addition*** to turning the first rotor.



The turning of the rotors means that how we encrypt a letter depends on when we encrypt a letter. For example, with the rotors in their starting position, the letter A will be encrypted as B. The first rotor will then turn, and so a new letter A would be encrypted as C.

2(a) [25 marks]

Write a program to simulate an Enigma machine, with the specified rotors and reflector.

Your program should first read in a line containing a single number *n* ($0 \leq n \leq 2^{31}$) which indicates the number of letters which have already been encrypted by the machine. You should then read in a second line which will contain a single word (only using the letters A, B, C and D) with between 1 and 10 (inclusive) letters.

You should output a single line containing the encrypted version of the word.

Sample run

```
14
AAABBB
```

`DBBDAD`

2(b) [2 marks]

If no letters have been encrypted so far, how would AAAAAA be encrypted?

For the following two questions you should suppose that you are able to choose how the rotors are wired. This means you can choose how the ports on the left of a rotor are wired to those on the right of the rotor; it is still necessary that each port on the left is linked to exactly one port on the right. The reflector cannot be rewired.

2(c) [4 marks]

Suppose there was only a single rotor. In how many different ways could it be wired so that, no matter how many letters are encrypted A will always be encrypted to B, B will always be encrypted to A, C will always be encrypted to D and D will always be encrypted to C? How about if there were two rotors? *[NB: Two wirings are different if the rotors' wiring in their initial positions is different.]*

2(d) [4 marks]

Suppose that, in addition to being able to choose how the rotors are wired, you are allowed more than two rotors. Is it possible that, with the machine in its initial state, if we encrypted an A it would become a B, but if we encrypted a B it would become a C? Justify your answer.

Question 3: ***Shirts***

Seven shirts, with the numbers 1 to 7 embroidered on the back, are hanging on a washing line. The order of the shirts can be changed by four different operations; in each case removing one of the shirts, pushing three of the shirts along the washing line to make a gap, and replacing the removed shirt back on the washing line in the gap:

1. The leftmost shirt is temporarily removed and the adjacent three shirts pushed left.
2. The rightmost shirt is temporarily removed and the adjacent three shirts pushed right.
3. The middle shirt is temporarily removed and the leftmost three shirts pushed right.
4. The middle shirt is temporarily removed and the rightmost three shirts pushed left.

For example, suppose that the shirts were on the washing line in the order `1234567`. The first operation would change the order to `2341567`, the second would change it to `1237456`, the third to `4123567` and the fourth to `1235674`.

Given the starting order of the shirts we are interested in finding the smallest number of operations that are required to change the order to `1234567`. For example, suppose the starting order was `3452671`. The smallest number of operations is 3 (i.e. applying operation 3 then 2 then 3).

3(a) [24 marks]

Write a program to determine the smallest number of operations to change a given order to `1234567`.

Your program should input a single line containing a seven digit number. This represents the starting order of the shirts and will containing each of the digits 1 to 7 exactly once. You should output a single number; the smallest number of required operations.

It is possible to get to the order `1234567` from any starting position.

Sample run

```
6417352
5
```

3(b) [3 marks]

From a fixed starting order, how many different orderings can the shirts be in after exactly 2 operations? How many after exactly 6? *[NB: This question asks that a certain number of operations are performed, not that a certain number of operations are required.]*

3(c) [4 marks]

The hardest starting order is the one which requires the largest number of operations. How many operations does it require? Give a starting order which requires this many operations.

3(d) [4 marks]

Suppose that you were able to choose both the starting and finishing orders. Is it possible to select orders so that more operations are required than the hardest case when only the starting position is chosen? Justify your answer.

Total Marks: 100 End of BIO 2008 Round One paper