British Informatics Olympiad

Time allowed: 3 hours

Sponsored by Jane Street

The 2023 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.
- 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: *Zeckendorf Representation*

In the *Fibonacci sequence* each number is generated by adding the previous two numbers in the sequence. We will start with the numbers 1 and 2, so the sequence is 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

The *Zeckendorf representation* of the number *n* consists of the distinct numbers from the Fibonacci sequence which sum to *n*, where no two adjacent numbers from the Fibonacci sequence are used. There is always a unique representation.

For example:

- 21 is represented by the single number 21;
- 21 is not represented by 8 13, even though they sum to 21, as those numbers are adjacent in the Fibonacci sequence;
- 100 is represented by 3 8 89.

The Zeckendorf representation for *n* always includes the largest number from the Fibonacci sequence that is no greater than *n*.

1(a) [24 marks]

Write a program that reads in a number (between 1 and 1,000,000 inclusive) and outputs the numbers (in any order) in the corresponding Zeckendorf representation.

Sample run

```
100
89 8 3
```

1(b) [2 marks]

What is the highest number under 1,000,000 whose Zeckendorf representation consists of a single number?

1(c) [2 marks]

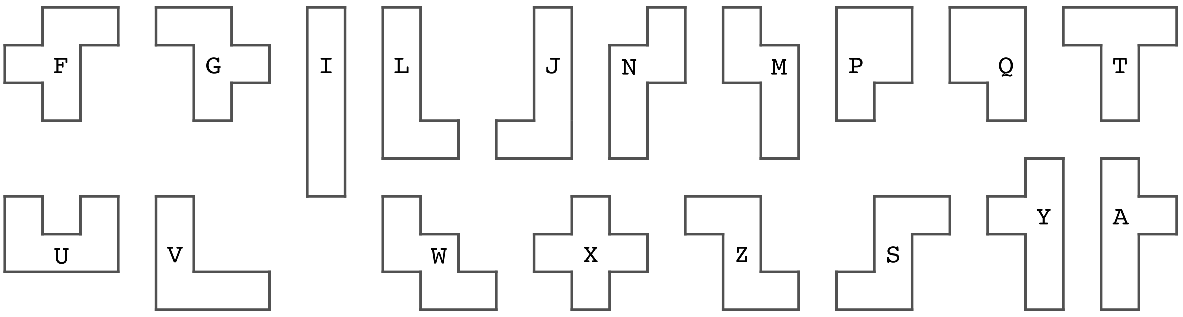
How many numbers under 53,316,291,173 have a Zeckendorf representation consisting of three numbers?

1(d) [4 marks]

How many numbers between 1,000,000,000 and 5,000,000,000 (inclusive) do *not* have 701,408,733 in their Zeckendorf representation?

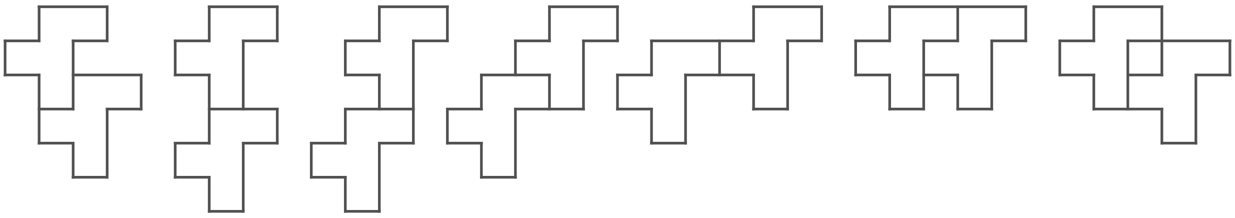
Question 2: *Pentominoes*

Pentominoes are the shapes that can be made when five equal sized squares are connected, with each square always edge-to-edge with another square. The number of different pentominoes depends on whether they are considered different when rotated and/or flipped over. We will consider the set of 18 pentominoes shown below (which are distinguishable if rotations only were allowed) although **throughout this question we will not permit flipping or rotations**.



Pentominoes can be combined to make more complicated shapes. When connecting pentominoes *at least one* square in each pentomino must be edge-to-edge with a square in a different pentomino.

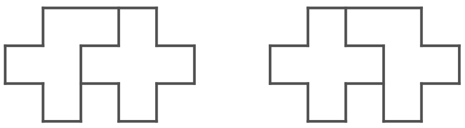
For example, two F pentominoes can be combined to make the following seven *distinguishable* shapes. These are the only combinations that can be formed because we are not permitting flipping or rotations.



Note that the last combination contains an *internal hole*. I.e. there is a gap inside the combined shape which is separated from the space outside the shape.

Two shapes are not distinguishable if, without flipping or rotation, the layout of their squares is the same, even if they were formed from different pentominoes.

For example, the following two shapes, generated from `FX` and `GX`, are not distinguishable:



2(a) [23 marks]

Write a program that calculates in how many ways a pair of pentominoes can be combined.

Your program should input a line containing a string of two capital letters, indicating the pentominoes to be joined. You should output the number of *distinguishable* shapes that can be formed by combining those pentominoes.

Sample run

```
FF
7
```

2(b) [2 marks]

How many shapes, made from combining `XW`, can also be made from two other pentominoes?

2(c) [5 marks]

How many distinguishable shapes can be formed by combining `III`? How about `LIV`?

2(d) [4 marks]

How many pairs of (potentially identical) pentominoes can be combined to form a shape with an internal hole?

Question 3: Dreaming Spires

In the *Tower of Oxford* game there are four pegs and four balls (numbered 1, 2, 3, 4) which can be stacked on the pegs. Each peg has a *height* of four, i.e. it can support a stack of all four balls.

We will denote a stack, running from the bottom to the top, by a sequence of digits running left (bottom) to right (top). E.g. `1234` is the stack with the balls in order and `1` on the bottom. An empty stack will be denoted by the digit `0`. The current *state* of the game will be shown by giving each peg's stack.

A valid move in the game consists of moving the top ball from a peg's stack and placing it on top of a (possibly empty) stack on another peg. Given a start position, the object of the game is to make the minimum number of moves to get to a given end position.

For example:

- Suppose the starting configuration has a single ball on each peg: `1 2 3 4`. There are 12 possible moves as each ball is on top of a stack and can be moved to any of the other pegs.
- The balls on the left-most two pegs could be swapped over by the following sequence of moves: `1 2 3 4` → `0 2 31 4` → `2 0 31 4` → `2 1 3 4`; i.e. by stacking ball `1` on top of ball `3`, then moving ball `2` to its final position ahead of ball `1` being moved.
- We could move from `1 2 3 4` to `2 3 4 1` using an equivalent swap sequence several times to swap `1` & `2`, then `3` & `1`, then `4` & `1`. This takes 9 moves.
- `1 2 3 4` → `12 0 3 4` → `12 3 0 4` → `12 3 4 0` → `1 32 4 0` → `0 32 4 1` → `2 3 4 1` requires only 6 moves.

3(a) [23 marks]

Write a program to determine the minimum number of moves required to get between two game states.

Your program should first input four numbers showing the initial state of the game. The next line of input will contain another four numbers giving the end state of the game.

You should output the minimum number of moves required to get from the start to the end state. It is possible to get between any two game states.

Sample run

```
12 0 3 4
1 32 4 0
3
```

3(b) [2 marks]

How many game states have 9 moves available?

3(c) [5 marks]

Suppose the starting configuration is `1 2 3 4`. How many different states can the game be in after 2 moves? How about after 4 moves?

3(d) [4 marks]

Consider a variant of the game where there are 8 pegs, 8 identical balls, and each peg has a height between one and eight inclusive. If there are 2023 different states of the game what are the heights of the pegs?

Total Marks: 100 End of BIO 2023 Round One paper