British Informatics Olympiad

Time allowed: 3 hours

Sponsored by Jane Street

The 2024 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: Integer Strings

A string is generated by joining together, in order, all the integers starting with *n*. We are interested in which *digit* appears in the i^{th} position in this string.

For example:

- If we start at 6 the string will begin 678910111213141516…;
- The 11^{th} digit in this string is 1 and the 12^{th} is 3;
- If we start at 999 the string will begin 999100010011002… and the 10^{th} digit is 0.

1(a) [25 marks]

Write a program that reads in integers *n* then *i* (both between 1 and 2^{59} inclusive), indicating the first number that appears in the string and the required digit from the string respectively. You should output the *digit* in the i^{th} position in the string.

Sample run

```
999 11
1
```

1(b) [2 marks]

Consider the string generated when *n = 1*. How many occurrences of the digit 5 appear in the first 101 digits?

1(c) [5 marks]

Consider the string generated when *n = 1*. The substring 123456789 first appears in this string between positions 1 and 9 inclusive.

Where does the substring 11111 first appear? Where does the substring 987654321 first appear?

Question 2: Parsing Lists

In this task you will manipulate lists of integers to create new lists. There are three *fundamental* lists that you will start from:

- `E` — the list of even integers, in order, starting with 2;
- `O` — the list of odd integers, in order, starting with 1;
- `T` — the list of integers, in order, where each one occurs the same number of times in the list as its value, starting 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, …

Given a list `L`, `L[i]` indicates the value in the i^{th} position of `L`. If `L` and `R` are two lists, `L` on the left and `R` on the right, when combined they are replaced by list `C` where the i^{th} position of C is `R[L[R[i]]]`. I.e. we find a value in list `R`, use it to find a value in list `L` and then use that value to find a final value in list `R`.

For example, if we combine `O` (on the left) and `E`:

- The 1st value in `E` is 2, the 2nd value in `O` is 3, and the 3rd value in `E` is 6;
- The 2nd value in `E` is 4, the 4th value in `O` is 7, and the 7th value in `E` is 14;
- The combined sequence is 6, 14, …

A *description,* giving details on the required manipulations, contains lists as well as brackets. Lists in the description will be combined until there is only a single list remaining. Where part of a description lies between brackets, it is treated as an independent description and that part will be manipulated (without combination with external lists) until reduced to a single list, at which point those brackets are then removed. When a description contains no brackets, the leftmost two lists are repeatedly combined forming a new description, until a single list remains.

For example:

- `E` is a description of a single list, so there is nothing further to do;
- `OE` involves a single combination leading to 6, 14, 22, 30, 38, …
- `E(OE)` will be the combination of `E` with `OE`, i.e. 94, 222, 350, …
- `EOT` is manipulated by first combining `EO`, then combining the resulting list with `T`, and is equivalent to `(EO)T`.

2(a) [25 marks]

Write a program that calculates the i^{th} value in a fully manipulated description.

Your program should input a string *d* of up to 12 characters (`E`, `O`, `T` or brackets) followed by *i* ($1 \leq i \leq 2^{60}$). You will not be required to access any value over 2^{61}.

You should output the i^{th} value in the fully manipulated description *d*.

Sample run

```
E(OE) 3
350
```

2(b) [2 marks]

How many 2s appear in `TT`?

2(c) [5 marks]

A *shortlist* of a list contains the first 100,000 entries of the list in order, and two shortlists are different if there is at least one position *p* where the p^{th} value of the two shortlists differs. How many different shortlists, generated from lists whose descriptions contain only brackets and at most 4 fundamental lists, contain the value 99?

Question 3: Word Game

In a *word game* letters are scored according to their position in the alphabet (`A`=1, …, `Z`=26) and a word's score is the sum of scores of its letters. Any combination of letters is allowed in a word, so long as there are no adjacent copies of the same letter.

Given a word, we are interested in finding its position in the alphabetically ordered list of words with the same score.

For example, there are 7 words which score 5. The word `BAB` is the 3rd word in the list, which is:

```
ACA
AD
BAB
BC
CB
DA
E
```

3(a) [25 marks]

Write a program to determine the position of a word in its list.

Your program should first input a string of 1 to 12 uppercase letters. The score of this string will be ≤ 75.

You should output the position of this string in the alphabetically ordered list of words with the same score.

Sample run

```
BAB
3
```

3(b) [2 marks]

What is the first word with the same score as `ACE`?

3(c) [4 marks]

How many words have a score of 26?

3(d) [5 marks]

What is the largest difference in word length between adjacent words in the list for a score of 54? How many adjacent pairs of words have this difference?

Total Marks: 100

End of BIO 2024 Round One paper