British Informatics Olympiad

Time allowed: 3 hours

LIONHEAD STUDIOS

The 2011 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 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. 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 *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: *Fibonacci Letters*

Each letter in the alphabet can be given a value based on its position the alphabet, A being 1 through to Z being 26. Two letters can produce a third letter by adding together their values, deducting 26 from the sum if it is greater than 26, and finding the letter whose value equals this result.

For example:

- A and B produce the letter C since 1+2=3 and A, B and C are respectively letters 1, 2 and 3 in the alphabet.
- P and Q produce the letter G since 16+17=33, 33-26=7 and P, Q and G are respectively letters 16, 17 and 7 in the alphabet.

We can generate a sequence of letters by starting with two letters and repeatedly using the last two letters in the sequence to produce another letter.

For example (starting with A and A) we have: A, A, B, C, E, H, M, U, H, C, …

1(a) [24 marks]

Write a program which reads in two capital letters (the 1st letter in a sequence followed by the 2nd letter) then an integer n ($1 \leq n \leq 1{,}000{,}000$). You should output a single capital letter, the n^{th} letter in the sequence that starts with the input letters.

Sample run 1

```
A A 7
M
```

1(b) [3 marks]

What letter together with F produces X? What letter together with Q produces H?

1(c) [3 marks]

Consider the sequence whose first two letters are both C. What is the 1,000,000,000,000,000,000th letter?

Question 2: *Accordion Patience*

Accordion Patience is a one-person game played with a normal *deck of cards*. Your task in this question will be to model various strategies for playing the game.

A deck of cards contains 52 cards. Each card has a *value* and a *suit* and there is one card of each possible combination in the deck. The suits are called Clubs, Hearts, Spades and Diamonds and the values in each suit are Ace, 2, 3, 4, 5, 6, 7, 8, 9, Ten, Jack, Queen and King. We will refer to the cards using the first letter of the value (or a digit) followed by the first letter of the suit. E.g. 4S for the "four of spades", TH for the "ten of hearts" and KD for the "king of diamonds".

Accordion patience is played by shuffling the cards, then dealing out all the cards into piles in a row from left to right. Initially all the piles will contain a singe card. Valid moves consist of taking a pile of cards and placing it on top of another pile of cards *to its left*. This can be done if the top cards of the two piles have either the same value or the same suit, and if the two piles are either next to each other in the row or separated by two other piles.

For example, suppose that we have piles with the following top cards 4S TH KD 4D. There are two valid moves, taking the pile with 4D on top and placing it either on the 4S pile (same value and separated by two piles) or on the KD pile (same suit and adjacent).

The game continues until either there is only one pile left (this is a *win*) or there are no more valid moves that can be made.

We will consider three different strategies for deciding which valid move to perform:

Strategy 1
Make the valid move using the right-most pile possible. If this pile can move to both an adjacent pile and a separated pile, move it onto the adjacent pile.

Strategy 2
Make the valid move that will create the largest pile possible. If there are several such moves, consider *only* those moves, and move the right-most pile possible (moving it to the adjacent pile if you still have a choice).

Strategy 3
Make the move that will leave the largest number of valid moves available next time. If there are several such moves, consider *only* those moves, and move the right-most pile possible (moving it to the adjacent pile if you still have a choice).

For example, suppose that we have the following piles:

	AD	8S	8D	4S	TH	KD	4D	TC
Number of cards in pile	1	3	1	2	1	2	2	1

By strategy 1 we will move the TC pile onto the TH pile. By strategy 2 we will move the 4D pile onto the KD one (the largest combined pile size is 4, the rightmost pile we can move to make this is the 4D pile and, since both moving it to the KD or 4S pile will create a pile of size 4, we pick the adjacent pile). By strategy 3 we will move the 8D pile onto the 8S pile which leaves 5 possible valid moves.

Initially the cards are ordered (from the top) with all the Clubs, then all the Hearts, then all the Spades then finally the Diamonds; with each suit ordered A, 2, …, 9, T, J, Q then K. They are shuffled by using six shuffling numbers *a, b, c, d, e* and *f* as follows: Cards are repeatedly taken, one at a time, from the top of the deck and placed on the bottom until the a^{th} card is reached; this is dealt to the row rather than placed on the bottom of the deck. This is repeated but waiting until the b^{th} card, then repeated for the c^{th}, d^{th}, e^{th} and f^{th}. We then repeat the process, starting with *a*. We continue until all the cards have been dealt.

For example, if the integers were all 1 the cards would be dealt in their original order and if the integers are 1, 3, 5, 7, 2 and 4 the cards would be dealt in the order:

AC, 4C, 9C, 3H, 5H, 9H, TH, KH, 5S, QS, AD, 5D, 6D, 9D, 2C, JC, KC, …, 7S 4H 3S

2(a) [24 marks]

Write a program that plays *accordion patience* by the different strategies.

You program should first read in six integers (each between 1 and 9 inclusive) indicating how the deck of cards should be shuffled. You should play the game three times, once for each strategy. Each time you play the game you should start with the cards in order, shuffle them according to the input, then play the strategy until you have won or cannot make another move.

The first line of your output should contain two cards, the left-most card after shuffling and dealing out the cards, followed by the right-most card. This should be followed by three more lines, the i^{th} of which containing information on the result of applying the i^{th} strategy — the number of piles that will be left after playing that strategy followed by the card on top of the left-most pile.

Sample run

```
1 3 5 7 2 4

AC 3S
6 9C
5 KC
4 4H
```

NB: The strategies will be marked separately.

2(b) [2 marks]

What would be the first 12 cards dealt if the six shuffling numbers were 2, 11, 3, 10, 4 then 9?

2(c) [4 marks]

Two sets of cards are said to be *different* if there is at least one card that is one set but not the other; the order of the cards does not matter. How many different sets can be generated by shuffling the cards, using six shuffling numbers between 1 and 9 inclusive, and taking the first six cards? How about if each shuffling number could be between 1 and 10 inclusive?

2(d) [5 marks]

Suppose a winning game of accordion patience has been played; i.e. all the cards finished in a single pile. That pile is now picked up and, without shuffling, used to deal another game of accordion patience; the top card of the pile being the left-most card dealt. Is it always possible to make at least one move? Justify your answer.

Question 3: *Upside-Down*

An *upside-down* number is an integer where the i^{th} digit from the left plus the i^{th} digit from the right is always equal to 10. For example 13579 is an upside-down number since 1+9 = 10, 3+7 = 10 and (since 5 is both the 3rd digit from the left and from the right) 5+5 = 10.

The first few upside-down numbers, in numerical order, are 5, 19, 28, 37, … , 82, 91, 159, …

3(a) [24 marks]

Write a program to determine the n^{th} upside-down number (in numerical order).

The input will consist of a single integer n ($1 \leq n \leq 2^{31}$). You should output a single integer giving the n^{th} upside-down number.

Sample run

```
11
159
```

3(b) [2 marks]

Consider all the different 9 digit numbers that use each of the digits 1, … , 9 once each. How many of these are upside-down numbers?

3(c) [3 marks]

How many digits are in the 1,000,000,000,000,000,000th upside-down number?

3(d) [6 marks]

Are there more upside-down numbers with 1000 digits that contain at least one 5, or more upside-down numbers with 1001 digits that contain at least one 5? Justify your answer.

Total Marks: 100

End of BIO 2011 Round One paper