DATA CONNECTION

BIO'96

The 1996 British Informatics Olympiad
Round One

Time allowed: 3 hours

For each question you are required to write a program for part (a), and produce written answers to the remaining parts. Programs may be used to help produce the answers to these written questions, and you may use a calculator.

Mark the first page used for your written answers with **your name**, **school/college**, and **age in years** when taking this paper. Number all pages in order if you use more than one sheet.

All programs must be saved, along with compiled versions if necessary, if they are to be counted. This includes programs used to help answer written questions. You should indicate clearly on your answer sheet what name you have used for each program. Marks for each part are given in square brackets by the questions.

An example run is given for parts 1(a), 2(a) and 3(a). **`Bold text`** indicates output/prompts from the program, and `normal text` shows data that has been entered. The output format of your programs should follow the 'example run' examples.

Question 1

Two numbers are said to be 'amicable' if they are different and the sum of the divisors of each number (including 1 but excluding the number itself) equals the other number.

For example: 2620 is divisible by 1, 2, 4, 5, 10, 20, 131, 262, 524, 655 and 1310; these add up to 2924. 2924 is divisible by 1, 2, 4, 17, 34, 43, 68, 86, 172, 731 and 1462; these add up to 2620. Therefore 2620 and 2924 are amicable.

1 (a) [20 marks]

Write a program which inputs two numbers (which will be less than 10,000) and then prints "Amicable" if they are amicable, or "Not amicable" otherwise. Your program should then terminate.

Sample run

```
First number: 2620
Second number: 2924
Amicable
```

1 (b) [3 marks]

Which is the lowest pair of amicable numbers?

Question 2

'Life' is a computer simulation played out on a rectangular board of squares. Each square has two possible states, on ('0' - zero) or off ('.'), and the state of the squares at the next generation is fully determined by the current state and some simple rules. The configuration at generation 0 is selected by the user.

Each square on the board has eight neighbours, those touching it directly (including diagonally), except border squares which have some of these pieces missing.

The rules for generating the next generation are:

Survival Any 'on' square with two or three neighbouring 'on' squares stays 'on' next generation.

Death Any 'on' square with less than two or more than three neighbouring 'on' squares goes 'off' next generation.

Birth Any 'off' square with exactly three neighbouring 'on' squares goes 'on' next generation. Otherwise it stays 'off'.

The rules stay the same for border squares, though they have fewer neighbours - in other words squares outside the 11x11 board are to be treated as permanently off. You should also remember that all births, deaths and survivals between generations occur simultaneously. For example:

Generation 0

```
..0..
...0.
.000.
.....
.....
```

Generation 1

```
.....
.0.0.
..00.
..0..
.....
```

Generation 2

```
.....
...0.
.0.0.
..00.
.....
```

2 (a) [25 marks]

You are to write a program to play Life on an 11x11 board. Your program should first read in a 5x5 board, which will be in the form of five lines of five characters (either '0' or '.'). This 5x5 board should be used as the centre section of the 11x11 board for generation 0, the rest of which is to be treated as off. This will be generation 0 for the entire run of your program. Once this is done you should display generation 0.

Until your program terminates you should repeatedly wait for input, and then:

1) If you receive an integer 'n' with a hash before it (such as #5), you should calculate generation 'n' and display it on the screen.

2) If you receive an integer 'n' with a plus before it (such as +5), you should calculate 'n' generations on from the last generation shown, and then display the new one on the screen.

3) If you receive the number -1 you should terminate.

4) Ignore any other input.

Sample run

```
.....
.....
...0.
....0
..000

...........
...........
...........
...........
...........
......0....
.......0...
.....000...
...........
...........
...........

#5
...........
...........
...........
...........
...........
...........
...........
......0.0..
.......00..
.......0...
...........

+1
...........
...........
...........
...........
...........
...........
...........
........0..
......0.0..
.......00..
...........

-1
```

2 (b) [4 marks] Why is it not feasible to have a '-n' option, which would calculate 'n' generations earlier?

2 (c) [3 marks] A 'rule set' for Life is the collection of Survive, Death, and Birth rules. Another rule set, for example, could have the same Survive and Death rules, but allow Birth with 3 or 4 adjacent 'on' squares. Note Survive, Death and Birth rules can only depend on the number of adjacent 'on' or 'off' squares. How many possible rule sets are there?

2 (d) [8 marks] Suppose we now limit rule sets so that the only valid Birth rules require "exactly 'n' neighbouring 'on' squares". Consider when generation 0 consists of the middle 5x5 squares all 'on', and all the other squares 'off'. How many rule sets produce, at generation 5, an 11x11 board with all squares 'off'?

Question 3 A domino is a 2x1 rectangle with 0, 1, 2, 3, 4, 5 or 6 written on each 1x1 half. A set of dominoes consists of the 28 unique dominoes. A domino with the same number on both halves is called a 'double'.

3 (a) [25 marks] You are to write a program that will fit a set of dominoes onto a grid which is 7 squares high, and eight squares across. This is to be subject to restrictions, in the form of doubles in fixed positions, and to the rule that two touching squares can only be equal if they belong to the same domino. A square should only be considered to touch the four squares directly above, below, left or right. Dominoes may not overlap.

You should first read in input indicating where the fixed doubles are. For each double this will be in the form of an x co-ordinate and y co-ordinate for one of the halves of the domino, and a R or B to indicate whether the other half is to the right or below. The top left corner of the board is at (1,1). The list of doubles will terminate with the number -1.

Note that not all the doubles may be fixed, and that their value is left for you to decide.

Once you have read the list you should try and fit the other dominoes to the grid. You should then output a valid solution if there is one, or print 'Impossible'. Your program should then terminate.

Sample run

```
8 3 B
4 1 R
5 7 R
-1

0-1 0 1-1 0-3 0
    |         |
5-0 2 0-6 1-2 4

1-3 1-4 1-5 1 0
            | |
3-2 4-2 5-2 6 0

2-6 3-4 3-5 3-6

4-5 4-6 5-6 5-5

6-6 3-3 2-2 4-4
```

3 (b) [4 marks] Suppose you are given some double restrictions, and a valid solution to the problem. Can you guarantee (or not) that there are any other solutions?

3 (c) [8 marks] Without any restrictions on doubles or the values on touching dominoes, how many ways can 28 dominoes be placed on the 7x8 board?

Total marks available: 100

End of the 1996 British Informatics Olympiad Round One Paper