The 1997 British Informatics Olympiad

Time allowed: 3 hours

You are required to 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. You may use a calculator and the on-line help that your programming language provides.

Mark the first page used for your written answers with **your name, age in years** and **school/college**. Label all your computer programs, and the floppy disk you use to submit the programs, with your name and school. Number all pages in order if you use more than one sheet.

For your programs to be marked, they must all be saved, along with executables if your language includes a compiler. This includes programs used to help answer written questions. You should also 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/prompts from the program, and `normal text` shows data that has been entered. The output format of your programs should follow the 'sample run' examples.

Attempt as many questions as you can. Marks allocated to each part are shown in square brackets next to the questions. Partial solutions (such as programs that only satisfy some of the specified requirements) may get partial marks. You may answer the questions in any order.

Question 1
Time

Given a time in numbers we can convert it into words. For example :

5:00	Five o'clock
5:10	Ten minutes past five
5:15	Quarter past five
5:30	Half past five
5:45	Quarter to six
5:47	Thirteen minutes to six

1 (a)
[20 marks]

Write a program which inputs two numbers (the first between 1 and 12, the second between 0 and 59 inclusive) and then prints out the time they represent, in words. You should follow the format of the examples above. Your program should then terminate.

Sample run

```
Hours: 4
Minutes: 12
Twelve minutes past four
```

1 (b)
[5 marks]

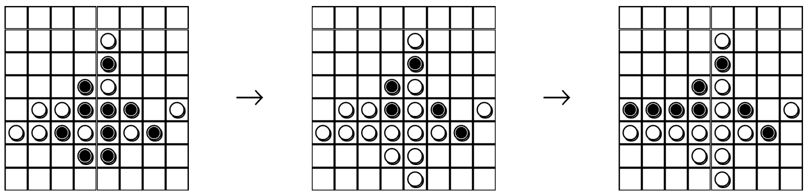
Which times, when written in words, have the longest length?

Question 2 ***Othello (Reversi)***

Othello is a game played on an 8 by 8 board with two players (White and Black). Each player has 32 pieces which are white (‘0’) on one side and black (‘*’) on the other. Play alternates between the two players, with White playing first.

A move is made by placing a piece, with the colour of the current player visible, on an unoccupied square of the board which has at least one of its eight neighbouring squares occupied. Any enemy pieces which are in a direct line between this piece and another of the player’s own colour (horizontally, vertically or diagonally) are turned over so that they show the current player’s colour. A piece may change colour several times during a game.

Example



The game ends when all the squares on the board are covered. The winner of the game is the player with the most of their colour showing.

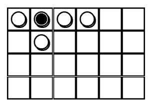
2 (a) [25 or 17 marks]

Write a program to play Othello. Both White and Black will be controlled by the computer, according to a fixed strategy. There are two strategies you may choose from. Implementing the first can earn you a maximum of 17 points for this question, whereas the second strategy is worth a maximum of 25 points.

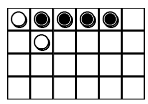
Strategy 1 Each player will always make the move which changes the largest number of pieces to their colour. If there are several equally good moves the one lowest down the board (on the screen) will be made, and if there are several of these the one furthest to the right will be chosen.

Strategy 2 Each player always considers the possible next move of their opponent. The move played is the one which guarantees that the largest number of pieces will be their colour after their opponent’s next move. With several equally good moves, the lowest/rightmost on the board is chosen as explained in strategy (1).

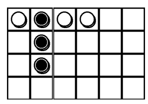
For example, if it is Black to play, and the board looks like:



Under the first strategy Black will play the move that turns over two White pieces:



Under the second strategy Black will play the move that turns over a single White piece:



White can now change the colour of at most one Black piece, leaving two on the board. If the first strategy’s move had been played White would be able to change the colour of all four Black pieces on the board, leaving none.

2 (a) cont'd

Your program should first read in a 4×4 board, which will be in the form of four lines of four characters – '*' for a black piece, '0' (zero) for a white piece, and '.' for an empty square. At least one of these squares will be non-empty. This 4×4 board should be used as the centre section of the 8×8 board the game is played on.

Once the central section has been input you should print which strategy your program is using, and display the board. Until your program terminates you should repeatedly wait for input, and then:

1) If you receive an integer n you should play the next n moves of the game, or as many as you can play before the game ends. You should then display the current state of the game.

2) If you receive the number 0 you should input another two numbers, representing an x co-ordinate and a y co-ordinate. You should then play a move (for the current player) in this position, and then display the board. The bottom left corner is (1,1).

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

4) Ignore any other input.

If playing a move takes you to the end of the game, you should print out which player has won and by how much (or print "Black and White draw."), and then terminate.

Sample run strategy 1

```
.0*.
0..0
0000
0*.0

Strategy 1
........
........
...0*...
..0..0..
..0000..
..0*.0..
........
........

1
........
........
...0*...
..0..0..
..0000..
..00.0..
....0...
........

0
7 4
........
........
...0*...
..0..*..
..0000*.
..00.0..
....0...
........

4
........
.....*..
...0**..
..0*.*..
.**00**.
..00.0..
....0...
........

0
1 4
........
.....*..
...0**..
..0*.*..
00000**.
..00.0..
....0...
........

-1
```

Sample run strategy 2

```
.0*.
0..0
0000
0*.0

Strategy 2
........
........
...0*...
..0..0..
..0000..
..0*.0..
........
........

99
********
******0*
**0***00
*000**00
*00*0000
*0*00000
**0**000
*****000

Black wins
by 8.
```

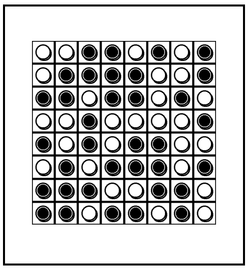
2 (b) [2 marks] What is the largest number of pieces (excluding the one being placed on the board) that can be turned over in one move?

2 (c) [5 marks] Suppose we start with the centre 4x4 squares all occupied. If it is White to play next, is there a strategy for Black that will ensure at least one corner is Black at the end of the game? Justify your answer.

2 (d)
[8 marks]

The following board position has been reached at the end of a game, and Black has just played. If you had been watching the last move there are 97 different games you may have observed. Black may only have played a piece on one of the 35 Black squares, but depending on the board's layout on the previous turn some of these moves may also have turned over additional pieces.

If you have been watching for the last two moves how many different games might you have observed?



Question 3
Egyptian fractions

When ancient Egyptians wrote fractions they were only able to use ones of the form 1/*a* – called unit fractions. An Egyptian wanting to write the fraction *b*/*c*, where *b* was not 1, had to write it as the sum of (different) unit fractions. All fractions *b*/*c* (*b* < *c*) can be written as an Egyptian fraction.

For example, the fraction 5/6 was written as 1/2 + 1/3, and 6/7 was written as 1/2 + 1/3 + 1/42. Egyptian fractions are not necessarily unique: 6/7 can also be written as 1/2 + 1/4 + 1/14 + 1/28. Since the unit fractions must be different, 2/3 can be written as 1/2 + 1/6, but not 1/3 + 1/3.

An Egyptian fraction representing *b*/*c* is 'minimal' if it uses the smallest number of unit fractions possible. An Egyptian fraction is 'optimal' if it is minimal, and the smallest unit fraction is as large as possible. There may be several minimal and several optimal Egyptian fractions for a given *b*/*c*.

For example 19/45 cannot be represented as the sum of two unit fractions, but there are 5 ways of representing it as the sum of 3 unit fractions: 1/3 + 1/12 + 1/180, 1/3 + 1/15 + 1/45, 1/3 + 1/18 + 1/30, 1/4 + 1/6 + 1/180 and 1/5 + 1/6 + 1/18. These five Egyptian fractions are all minimal. Only 1/5 + 1/6 + 1/18 is optimal, since 1/18 is larger than 1/30, 1/45 and 1/180.

3 (a)
[23 marks]

Write a program that inputs two numbers *a* and *b*, and calculates an Egyptian fraction representing *a*/*b* with no more than *a* unit fractions. You will only be given fractions with 0 < *a* < *b* < 1000. Your program should then output the denominators (bottom parts) of the unit fractions.

Sample run

```
Numerator: 19
Denominator: 45
5 6 18
```

There will be ten tests. For each test, if there exists an optimal solution which is the sum of 4 (or fewer) unit fractions, more marks will be given for outputting an optimal solution.

No unit fraction will need to be smaller than 1/32000. [You can still output denominators greater than 32000, so long as they are no more than 8 digits long.]

3 (b)
[2 marks]

What fractions do the following Egyptian fractions represent?

(i) 1/8 + 1/56

(ii) 1/2 + 1/4 + 1/8 + 1/16 + 1/32

3 (c)
[5 marks]

Do there exist any fractions for which there is a unique Egyptian fraction?

3 (d)
[5 marks]

How many different valued fractions (*a*/*b*) are there if 0 < *a* < *b* < 1000?
[Hint : Remember, for example, that 1/2 and 2/4 have the same value.]

Total marks available: 100

End of the 1997 British Informatics Olympiad Round One Paper