2010 SCHOLARSHIP EXAMINATION

PRACTICAL SECTION

The University of Waikato
Te Whare Wānanga o Waikato

DEPARTMENT	Computer Science
COURSE TITLE	Computer Science Scholarship
TIME ALLOWED	Six hours with a break for lunch at the discretion of the supervisor
NUMBER OF QUESTIONS IN PAPER	Three
NUMBER OF QUESTIONS TO BE ANSWERED	Three
GENERAL INSTRUCTIONS	Candidates are to answer ALL THREE questions. All questions are important. Answer as much of each question as you can. Plan your time to allow a good attempt at each question, but be aware that Question 3 is the most difficult and will take considerably longer than the others.
SPECIAL INSTRUCTIONS	Please hand in listings, notes and answers to written questions, and a CD/DVD with your program/computer work for each question. Please make sure that a copy of each program is printed, or stored as a plain text file. You cannot assume that the examiner has available any special software that might be required to read your files. Candidates may use any text or manual for reference during the examination.
CALCULATORS PERMITTED	Yes

TURN OVER

1. **What if?** (Spreadsheet Use)

In this question you are asked to use a spreadsheet to do calculations and to display the results. We expect that the spreadsheet will be used for all calculations unless the question states otherwise - you will be marked down for performing calculations by hand and directly entering the results. Your work will be graded on three criteria.

(i) The accuracy of your results.

(ii) The skill you show in making use of the capabilities of the spreadsheet.

(iii) The presentation of your results. We have deliberately not provided any instructions concerning layout or formatting and example graphs may lack labels and proper scales.

Years of study were finished; a degree majoring in Computer Science achieved with minors in Video Production and English Literature; many hours spent playing computer games. It was now or never. The group of four friends had financial support and agreed that they would live cheaply, work in one of their parent's basements, and devote their time to developing the greatest computer game yet written.

Your task is to do the financial analysis to reassure their financier that they have a good chance of making a large profit. Financial details are as follows:

- The project begins at the start of January 2011. You should calculate financial results monthly for 10 years (until December 2020).
- The four friends will each receive salaries of $1000 per month, for the months taken to develop the game. After that time they will receive no salary (but may share in profits).
- The money for salaries will be borrowed from the financial backer and paid to the friends at the start of each month.
- The project will be charged interest on the outstanding loan at the end of each month. The interest charged will be at the rate of 15% per annum (per year) on the balance of the loan as at the end of the prior month. Interest costs will be added to the outstanding loan.
- All money made from selling the game will go to repayment of the loan, until it is fully repaid. Further money earned will be all profit.

For example: In the first month (Jan 2011), $4,000 is borrowed, there is no interest payment, and no sales, so the project owes the backer $4,000 at the end of the month. At the end of Feb 2011, a further $4,000 has been borrowed, there is an interest charge of $50 (at 15% per annum on the balance at the end of Jan), so the total loan reaches $8050.

It is expected that the game will take 12 months to develop. Sales will begin in the following month. The friends estimate that sales of their game will be 100 in the first month and increase at 10% per month after that (110 in the second month, 121 in the third, etc). The game will be sold at $49.95 per copy. For the purposes of your simulation, you can assume that there are no other costs.

If the rate of sales continues to grow at 10% per month for a long time, profits could be astronomical. However, the friends are realistic. No matter how good their game, eventually someone will release something better. On the month that a better game is released by competitors, they estimate that their sales will drop to 100 for the month, and after then decline at 5% per month until no more are sold.

(Question 1 - continued on next page)

You should proceed as follows in answering the question.

(a) Build a spreadsheet which calculates the total amount of money borrowed over the period January 2011 to December 2020. You should start with the salary payments and then add the interest charges.

(b) Extend your program to calculate the number of copies of the game sold and the total income.

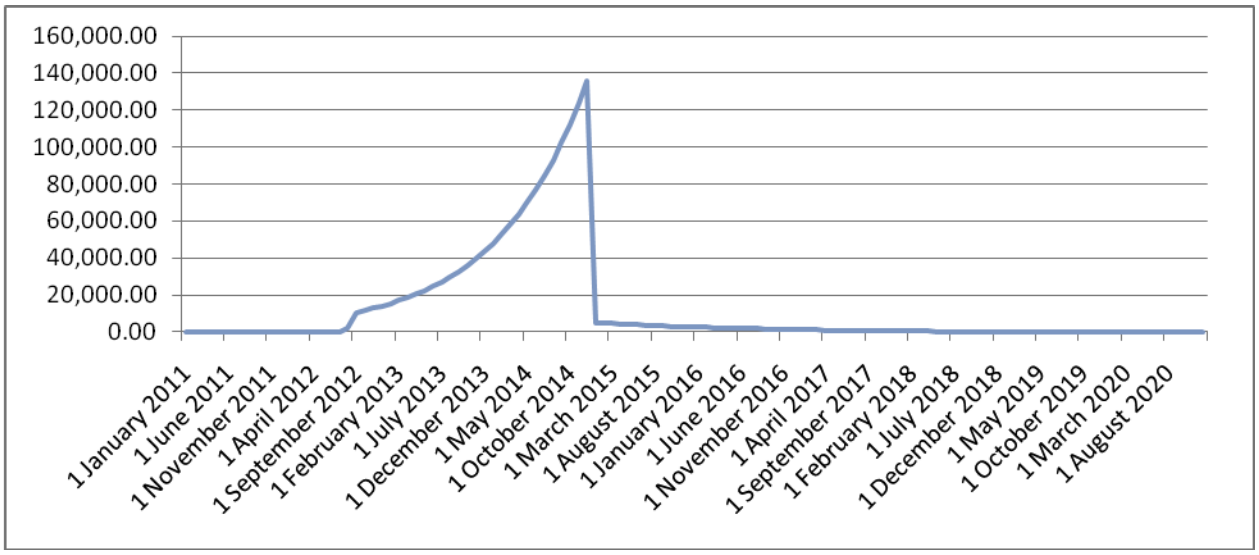
(c) Extend your loan calculations to allow for repayment using income from game sales.

(d) Calculate the monthly profit (if any) over the period Jan 2011 to December 2020 (inclusive).

(e) Add a graph (chart) of monthly profit against date.

(f) Make sure your spreadsheet clearly shows the total profit (if any) earned over the period of the project

The financier is particularly worried about the release date of a successful competing game. In the sample graph shown below, that occurred in January 2015. If it had occurred later, larger profits would have been made. If it occurred early enough there might be no profit at all. You should set up your spreadsheet to make it easy to experiment with this date and clearly see the consequences of any change.



Roughly formatted graph of monthly profit in dollars (after the loan has been repaid), with a competitor's game being released in January 2015.

2. **A question of age** (Careful and Accurate Programming)

Your programming work in this question will be assessed on two criteria:

(a) Completeness and accuracy of the program.

(b) Good presentation. That is, it should make good use of programming language facilities, be well organised, neatly laid out, and lightly commented.

Note also that this question also asks for a written answer to one question. You may write this by hand (recommended), or use a word processor and include the file with your results.

It was at a political meeting. A candidate was trying to explain to the audience that the population of Hamilton is aging (ie: because people are living longer and the birth rate is dropping, the average age of the population of Hamilton is increasing).

"Just look around the room", she said. "What do you think is the average age of the people here? When I first stood for council in 1967, I asked the same question at a meeting, because I was arguing that the city needed to improve its amenities for young people. I'll tell you the answer I found then, but first let's answer it here for this group."

Everyone looked around, and people came up with different estimates: "38", "45", "42". How could they work out their average age? The difficulty is that most people don't like admitting their ages in public.

Fortunately the candidate had a clever solution. She produced a calculator, tapped in a large number and handed the calculator to the first person in the room. That person, and then each other person in turn added their age to the number on the screen. After everyone had added their age, the candidate subtracted the number she had first entered and divided by the number of people. The result was the average age – and no-one had been required to reveal their age to the others. Given this vital information the meeting continued, but we can leave it there and concentrate on the method used to obtain the average age.

Of course, as a computing expert, you will appreciate the possible flaws in this method. Even if no-one tries to cheat it is still possible that:

- People might make mistakes – perhaps hitting a key twice – so a 23 year old might enter 223.
- Someone might press the wrong key and lose the whole calculation.
- The candidate might forget the first number, or get it wrong in the final calculation.
- Someone might be missed, giving an inaccurate count for the number of people.
- If there are only two people in the room, there is no way of presenting an average without each person discovering the age of the other.

Your task is to write a program to improve the process. You will write it on the computer that you are using for the exam, but imagine it being used on a small hand-held computer in much the same way as the calculator was used. Of course, your program could simply ask people to enter their ages and just show the final answer, but there was something that the people at the meeting liked about seeing the current total, and watching their age being added, so you have been asked to follow the same basic method. From one person's point of view the interaction should look something (but not exactly) like this:

```
Current total:   7845
Enter your age:    45
New total:       7890
Press <Enter> to clear the screen

Current Total:   7890
…
```

(Question 2 - continued on next page)

(a) Write a program to be used for this average age calculation. You are free to alter the interaction as you see fit.

(b) Write an explanation of the way in which your program avoids the flaws shown for the calculator method, and any other potential difficulty you have considered.

You can assume that all the people in the room are honest – no-one will deliberately enter false information; but they are all curious about the other people's ages and will try to determine them if they can.

TURN OVER

3. **Queen and pawns** (Problem Solving and Programming)

Your programming work in this question will be assessed on two criteria:

(a) Your approach to the problem. We will be looking at your work for evidence that you found good ways of storing the necessary data, and devised algorithms for finding and displaying the requested results. Please hand in any notes and diagrams which describe what you are attempting to program, even if you don't have time to code or complete it.

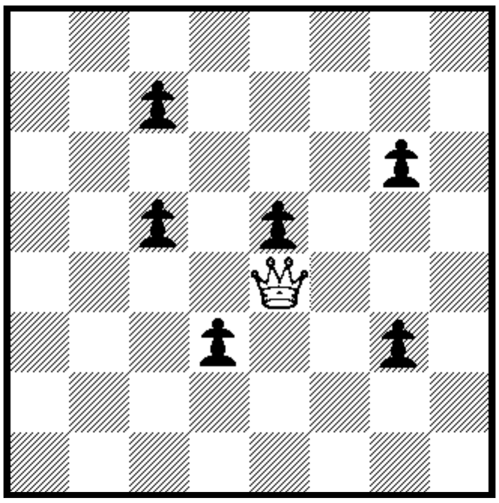
(b) The extent to which your program works and correctly solves the problem.

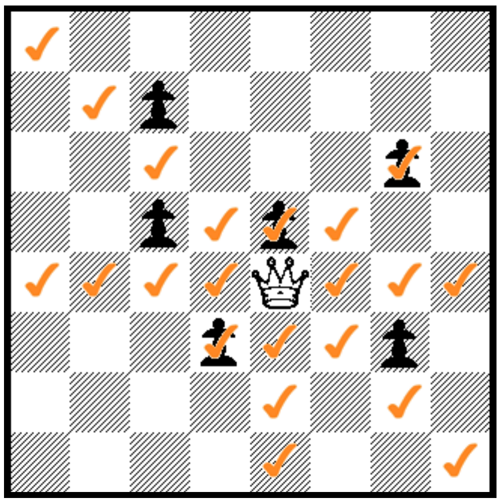
This problem asks you to work out whether certain moves on a chess board are legal. Do not worry if you are not familiar with the game of chess. There should be sufficient explanation here to enable you to solve the problem

The game of chess is played on an 8 by 8 board of alternating black and white squares. In the full game of chess each player has 16 pieces selected from 6 different kinds. One player plays with black pieces and one with white. Our problem involves only two kinds of piece: black pawns ♟ and a white queen ♕

In our problem some number of black pawns is placed on the board. You are then given a position for the queen and a position it might move to. Your task is to decide whether the queen can legally make the move. Under the rules of chess, a queen can move for any distance in any straight line along a row or column or diagonal of squares, so long as the path over which it passes is clear of other pieces (in our case pawns), and it stays on the board. It is legal for the queen to land on a square holding a pawn – in that case the pawn is said to be 'captured'.

For example with a placement as shown on the left below, the queen may move to any of the squares ticked on the right:





(Question 3 - continued on next page)

Proceed as follows

(a) Write a program to read a description of a layout of pawns on a chess board. One way of entering the data is to begin with a number to say how many pawns are on the board, and follow it with two numbers for each pawn giving its row (1 to 8 from top to bottom) and column (1 to 8 from left to right). For the board shown these numbers would be entered:

```
6
2 3
3 7
4 3
4 5
6 4
6 7
```

(b) Display your chess board on the screen. In text form it could look like this

```
+---+---+---+---+---+---+---+---+
+   +   +   +   +   +   +   +   +
+---+---+---+---+---+---+---+---+
+   +   + P +   +   +   +   +   +
+---+---+---+---+---+---+---+---+
+   +   +   +   +   +   + P +   +
+---+---+---+---+---+---+---+---+
+   +   + P +   + P +   +   +   +
+---+---+---+---+---+---+---+---+
+   +   +   +   +   +   +   +   +
+---+---+---+---+---+---+---+---+
+   +   +   + P +   +   + P +   +
+---+---+---+---+---+---+---+---+
+   +   +   +   +   +   +   +   +
+---+---+---+---+---+---+---+---+
+   +   +   +   +   +   +   +   +
+---+---+---+---+---+---+---+---+
```

(c) Accept four numbers, being the row and column for the current position of the queen and the row and column of the position it might move to. Your program should decide whether the move is legal. If so, it should say ‘Legal’. If not it should say ‘Illegal”. Note that the queen’s starting square should not be one that holds a pawn. You will find it helpful to display queen positions on screen as you test your program. That might be done by extending your display from part (b).