THE UNIVERSITY OF WAIKATO

COMPUTER SCIENCE SCHOLARSHIP EXAMINATION

2007

PRACTICAL SECTION

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/floppy disk with your program/computer work for each question. Please make sure that copies of programs are stored as plain text files. You cannot assume that the examiner has available any special software that might be required to read your files. Candidates may use any texts or manuals for reference during the examination.
CALCULATORS PERMITTED:	Yes

1. **Currency Exchange (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 - you will be marked down for performing calculations by hand and directly entering the results. Your work will be graded on three criteria.

(a) The accuracy of your results.

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

(c) The presentation of your results. We have deliberately not provided any instructions concerning layout or formatting

The foreign exchange rate for the New Zealand dollar (NZD) has varied greatly against the United States dollar (USD) during 2007. The web site ExchangeRate.org has provided us with a list of the USD-NZD daily exchange rates for the period from March 28, 2007 to September 24 2007. The list is stored on the CD as a plain text file, where each line has three fields separated by commas. These are the:

date, day-of-the-week, purchase price of one USD in terms of the NZD.

For example, the first three lines of the file are as follows:

```
28/03/2007,Wednesday,1.40706
29/03/2007,Thursday,1.39958
30/03/2007,Friday,1.39704
```

Your task is to build a spreadsheet using this data to satisfy the instructions following. [Note that your spreadsheet should recalculate without human intervention if any of its data fields are changed.]

(a) Load the data into a spreadsheet. As you will see from the example above, the data is supplied as comma separated values with New Zealand formatted dates. You will need to take care when you import the file into your spreadsheet that you specify this format (depending on how your computer is setup, it may expect dates in US format – month, day, year).

(b) Plot a graph of exchange rate against time. Use appropriate x and y axis labels and a caption.

(c) Find the minimum and maximum exchange rate.

(d) Display true in a cell if there were any days other than Sundays where the exchange rate was the same as the day before. Display false otherwise.

(e) Imagine you have been asked to predict a day's exchange rate from the data for the days immediately before it (without using any other information). A naive approach is to use the value from the day before as the prediction for the day's rate. If this is done with the data given, the average difference between the prediction and the actual value (excluding Sundays where the prediction is always exactly right) is 0.01046. Say why you think you can, or can not, do better than this naive approach? Design, implement and test an algorithm (that is different to the naive algorithm) that you think might give a good result. Please include a written explanation of your algorithm with the material you hand in. (NOTE: do not try more than one algorithm).

Please print a partial listing of your spreadsheet and graph to hand in. As the data set is very large you should not print the entire spreadsheet.

2. **<u>All Black Selection (Careful and Accurate Programming)</u>**

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.

A local radio station is running a competition to see which of their listeners can get closest to guessing the team the All Blacks will field for an upcoming match. Given a list of players in the squad, listeners must decide not only which players will be selected to play, but also which jersey number they will wear in the game.

Each player is assigned a squad number. Here are the New Zealand squad numbers for the Rugby World Cup 2007:

Jerry Collins(1), Carl Hayman(2), Andrew Hore(3), Chris Jack(4),
Sione Lauaki(5), Richie McCaw (Captain)(6), Chris Masoe(7), Keven Mealamu(8),
Anton Oliver(9), Keith Robinson(10), Greg Somerville(11), Rodney So'oialo(12),
Reuben Thorne(13), Neemia Tialata(14), Ali Williams(15), Tony Woodcock(16),
Daniel Carter(17), Andrew Ellis(18), Nick Evans(19), Doug Howlett(20), Byron Kelleher(21),
Brendon Leonard(22), Luke McAllister(23), Leon MacDonald(24), Aaron Mauger(25),
Malili Muliaina(26), Josevata Rokocoko(27), Sitiveni Sivivatu(28), Conrad Smith(29), Isaia Toeava(30).

The actual team selected by Graham Henry to play France in jersey order (1 to 15) was:

Woodcock, Oliver, Hayman, Robinson, Williams, Collins, McCaw (capt), So'oialo,
Kelleher, Carter, Sivivatu, McAlister, Muliaina, Rokocoko, McDonald.

Making use of the squad numbers this can be encoded from #1 to #15 as:

16, 9, 2, 10, 15, 1, 6, 12, 21, 17, 28, 23, 26, 27, 24

Radio station listeners will text their selections encoded in this manner.

Your task is to compute the match between a listener's selection and the real team selected by Graham Henry. You should develop your program in two stages.

<u>Stage A.</u>

Write a program to score a selection as follows:

A score of 1 point is achieved if you get the correct player in the correct jersey number, 0 otherwise.

For example: The team encoded as 16, 8, 11, 4, 15, 1, 6, 12, 21, 17, 27, 23, 30, 20, 26 scores 8 points.

<u>Stage B.</u>

Modify your program to score in the following way:

2 points for correct player in the correct jersey, 1 point for correct player in the wrong jersey and 0 otherwise.

This time the team encoded as 16, 8, 11, 4, 15, 1, 6, 12, 21, 17, 27, 23, 30, 20, 26 scores 18 points as follows:

2 points for (Woodcock, Williams, Collins, McCaw, Sooialo, Kelleher, Carter, McAlister) = 16
and 1 point for (Rokocoko, Muliaina) = 2, making a total of 18.

3. **Tic-Tac-Toe (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.

Overview

Tic-Tac-Toe (a.k.a. Noughts & Crosses) is a game played by two people, where each takes a turn placing his/her mark in one of the empty spaces in a 3-by-3 grid (one player marking with an X, the other with an O). If either player succeeds in placing three of their marks in a straight row (across, down, or on a diagonal) then they win; otherwise the game ends in a draw. The simplicity of Tic-Tac-Toe makes it a good game for introducing computer science students to the field of artificial intelligence and automated reasoning.

Objective

Your task is to implement Tic-Tac-Toe as a computer program that allows a human player to compete against the computer. You must create a suitable interface which supports the following:

i. A representation for the game grid

ii. A means for both the human player and computer to take turns making their mark in an empty space,

iii. A way to determine when the game is over and who (if anyone) has won, and

iv. A method to keep score over a series of repeated games
(i.e. a record of how many games each player has won).

For the first game, the player to go first should be chosen at random; but thereafter players should alternate as to who goes first.

Assessment

The 'beauty' of your user-interface will be a low-priority in the assessment of your solution. More or less anything that supports the functionality is appropriate. Of greater interest for the examiners is the overall design, efficiency, flexibility and intelligence of your solution.

For example, you may start out having the computer play naively by choosing its moves at random (from the available spaces); but as time permits you will want to make it select moves more 'intelligently'. You may want to incorporate some or all of the tactics from the following strategy for playing perfect Tic-Tac-Toe:

1. Win: complete three in a row.
2. Block: block their opponent from completing three in a row
3. Fork: threaten a win with two possible completions in two ways
4. Block Fork 1: if there is a configuration where the opponent can fork, create two in a row to force a block
5. Block Fork 2: if there is a configuration where the opponent can fork, block that fork
6. Centre: play the centre
7. Opposite Corner: if the opponent is in the corner, play the opposite corner
8. Empty Corner: play an empty corner
9. Empty Side: play on an empty side

QUESTION 3 CONTINUED ON NEXT PAGE

QUESTION 3 CONTINUED

Advanced solution

Although Tic-Tac-Toe is traditionally played on a 3-by-3 grid, it is also sometimes played on a 4-by-4 grid, and could be played on a 5-by-5 grid, or a 6-by-6 grid, and so on (although playability declines rapidly as the dimension of the game increases).

Implementing a good solution for the 3-by-3 game is satisfactory for this examination; but extra credit will go to a solution that works for an N-by-N game grid (where N is any positive integer between 3 and 9, inclusive) simply by setting the value for N as a parameter to the system. That is, the value for N is defined in just one place (either at compile-time or run-time) and the same code will play the game correctly on the corresponding N-by-N grid.

Please hand in printed listings of your programs, and a version of each program on a CD or floppy disk. The version on CD/floppy disk should include a plain text version. You should also hand in any notes, diagrams or explanations you have written. These are particularly important if you do not get your program running correctly as they give us information about your method of approaching the problem. If your program generates output to a file, or to a printer, you should also include printed version of the output.