THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2008 COMPUTER SCIENCE SHOLARSHIP EXAMINATION

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

1. **Peak Oil (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.

(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

Studying oil production in the 1950's M. King Hubbert discovered that the rate of production of an oil field over time generally follows a bell shaped curve. Production starts at a low level with the first well drilled; it increases at an accelerating rate as more wells are added; then the rate of increase slows until 'peak production' is reached. After that production drops, following a curve that is the mirror of the earlier growth. An equation for the bell curve is

$$Pe^{-(t-\mu)^2/\sigma^2}$$

Where t is time (in years), P is the maximum (peak) rate of oil production, μ is the year of peak production, and σ sets the time scale. Small σ (in years) means that the whole process is quite quick. Larger σ means that it takes longer time.

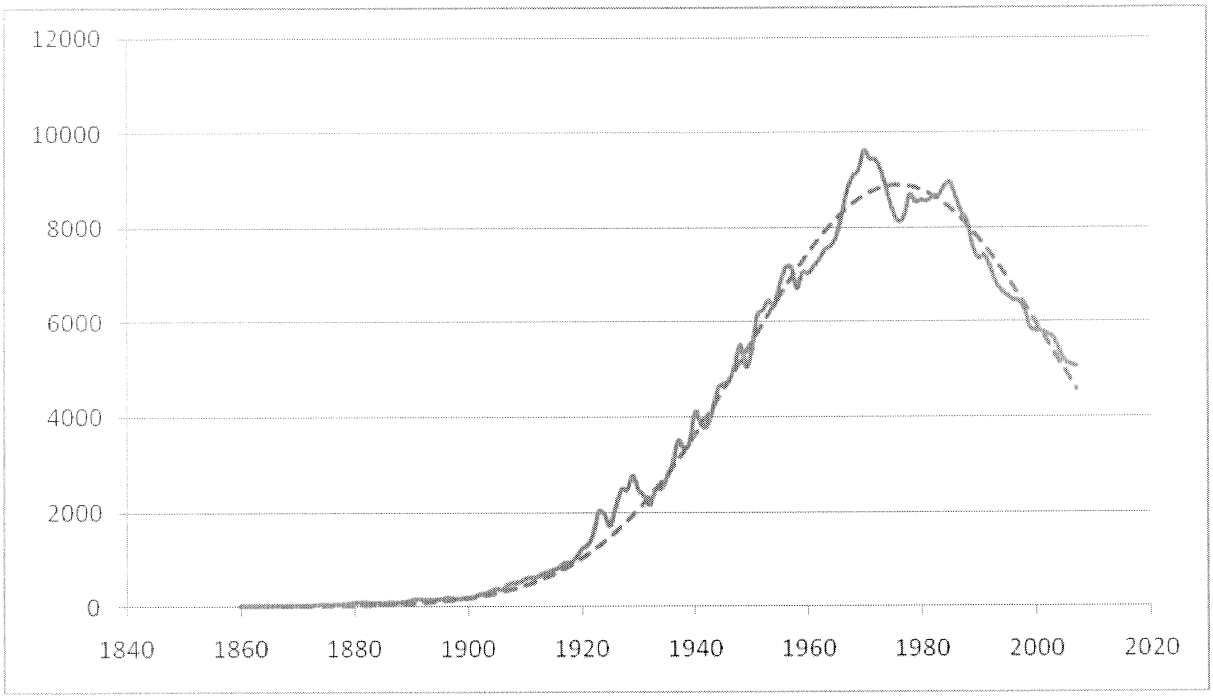
(1) The accompanying CD has a file USOil.csv holding production information for the US oil industry from 1860 to 2007. Each line of the file has a year and a production volume for that year. The production volume is in 1000's of barrels.
(2) Create a spreadsheet holding the US oil production data.
(3) Insert a graph showing US oil production over the given time period (see next page for a sample graph).

We could use a sophisticated mathematical approach to try fitting this data to a bell curve; but the spreadsheet offers an alternative approach – we can set up a calculation and choose P, μ and σ by trial and error.

(4) Set up cells to hold P, μ and σ values.
(5) Add a column to calculate the bell curve, based on the values in the P, μ and σ value cells. Hint: To calculate e to the power x in a spreadsheet, write exp(x)
(6) Modify your graph to show both the US oil data and the calculated bell curve together.
(7) Experiment until you have P, μ and σ values that give a good match to the oil data. Note that the match is not exact – the actual oil data only approximately follows a bell curve.

(Question 1 - continued on next page)

Your graph should look something like this



(8) Extend your graph so that you can read off the year in which US oil production will have dropped to 25% of its peak value. To do this you will have to extend the range of dates for which you calculate the bell curve and extend the horizontal axis of the graph to show the new values.

The file NewField.cvs holds data up to 2007 from a little known new oil field opened in 1955. Although the amount of data is quite small, we can still fit a bell curve to it.

(9) Start a new spreadsheet. Load the NewField.cvs data.

(10) Fit a new bell curve and experiment to predict the year of peak production, and the peak production quantity of the new oil field.

The US oil production data is taken from http://tonto.eia.doe.gov/dnav/pet/hist/mcrfpus2a.htm (provided by the US Energy Information Administration)

2. **Desk Calculator (Careful and Accurate Programming)**

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

(a) Completeness and accuracy of the program.

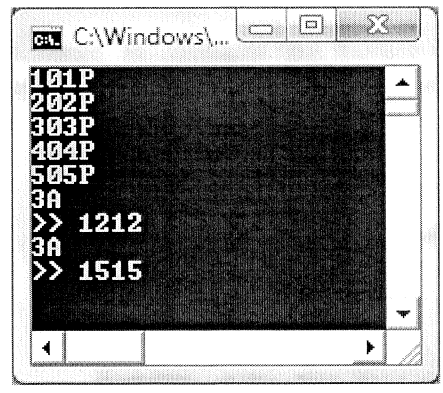
(b) Good presentation. That is, it should make good use of programming language facilities, be well organised, neatly laid out, and lightly commented.

Your problem is to write an unusual kind of desk calculator. It should take instructions from the keyboard and display results in a console window. Please read through the question carefully to make sure that you understand how the calculator should operate

The calculator works on a list of numbers. Numbers can be entered using the 'Push' command or can be left by a calculation. Arithmetic commands can be used to add, subtract, multiply or divide some of the most recently added numbers in the list. All commands take the form of a number followed by a letter. The commands are Add, Subtract, Multiply, Divide, Push and Quit. In the case of the push command, the number is the value to be put into the list. In the case of the arithmetic commands the number tells the program how many items to take off the list. Those items then have the arithmetic operation performed on them, and the (single number) result is put back onto the list. The commands are illustrated in the following demonstration. Output from the program is underlined.

Input/Output	Explanation
101P	Push puts the number 101 into the list
202P	Push 202 into the list
303P	Push 303 into the list
404P	Push 404 into the list
505P	Push 505 – at this stage there are 5 numbers in the list
3A	Take the last 3 numbers off the list (gets 303, 404, 505),
>> 1212	... adds them, puts the result on the list and displays it
	... leaving the list with (101, 202, and 1212)
3A	Add 101, 202 and 1212, push the result onto the list
>> 1515	... and display it (1515)
303P	Push 303. Now the list has two numbers (1515 and 303)
2D	Divide the last two numbers on the list (1515 / 303 = 5)
>> 5	... again displaying the result
101P	Push 101
5P	Push 5 giving a list with 5, 101 and 5
3M	Multiply the three numbers
>> 2525	... and display the result
5P	Push 5
101P	Push 101, giving a list with 2525, 5, 101
3D	Divide the three numbers on the list
>> 5	... and display 2525 / 5 / 101 => 5
0Q	Quit

In the console window the first few commands look like this



3. **Battleships (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 that 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.

In this problem you are asked to work towards building a computer form of the popular children's game 'Battleships'. Building a computer game can involve a great deal of work by many people. A good game will have high quality graphics, a well designed user interface and be capable of playing 'intelligently' (the so called AI or 'artificial intelligence' component of game development). Your task is to work on AI for the 'Battleships' game. You will do this by building enough of a game program to test its playing strategy – you will not build the entire game.

A typical set of instructions for the battleships game, taken from a game website, is shown on the next page. Many aspects of the game can be varied: the size of the grid, number and types of ships, ways of scoring, etc.

(1) Decide on a way of storing a game grid. Write a program to hold a game grid and display it on the screen. This DOES NOT need sophisticated graphics. Simply displaying a grid with text is fine. Here is a sample grid showing an aircraft carrier starting at B2, and a cruiser at D3:

```
A   . . . . . . . .
B   . A A A A A . .
C   . . . . . . . .
D   . . C . . . . .
E   . . C . . . . .
F   . . C . . . . .
G   . . . . . . . .
H   . . . . . . . .
    1 2 3 4 5 6 7 8
```

(2) Extend your program to accept input from the keyboard to allow a user to enter ships onto the game grid. Allow just two kinds of ship: aircraft carriers (which occupy 5 cells in a horizontal or vertical row) and cruisers (3 cells).

For experiments with strategy (AI) we will play a modified version of the battleships game. A player has a fixed number of turns (eg: 20). They get a point for every hit on a ship square. (Remember that to sink a ship we have to hit every square it occupies, so it is possible to get 5 points hitting one aircraft carrier.) The goal is to get as many points as possible. A strategy is a way of choosing squares to shoot at. It is possible to test a strategy with just one player shooting. It is not necessary to implement the full game with two players.

(3) Modify your program to test playing strategies, e.g. generate 20 shots and score them.

(4) Write down any ideas you have for playing strategies. Implement and test some or all of these. Describe your results.

Instructions for Battleships (from http://www.activityvillage.co.uk/battleships_instructions.htm)

Give each player a pencil and a print-out of the Battleships game. The top grid is for your own fleet ("My Ships") and the bottom grid is where you try to locate the other player's fleet ("Their Ships").

First you decide where to place your own fleet within your grid. A fleet is made up of one Aircraft Carrier, one Battleship, one Cruiser, two Destroyers and two Submarines. Each type of ship covers a different number of boxes in the grid, as shown on the print-out, and is drawn vertically or horizontally (not diagonally). Ships cannot occupy the same square.

To place a ship, check how many boxes are covered by the ship (shown to the left of your grid) and then write the first letter of the name of the ship in the boxes it covers. For example, a Cruiser covers three boxes so you would pick any three adjacent boxes and put the letter C in each box. Keep your fleet location secret from your opponent! When each player has marked their fleet on their grid, begin play.

Take turns to "shoot" at your opponents' fleet by calling out the number of a certain box by its grid location. For example, you could call out "B4" or "D1". Your opponent must say whether the shot is a "miss" or a "hit", and, if it is a "hit", what type of ship it is. You can keep track of what you have shot on your lower grid, and the ships you have sunk by crossing off the ships at the bottom right of your print-out. To sink a ship you must shoot all of the squares it occupies.

Play continues until one player wins by successfully sinking the whole of the other player's fleet.