THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

2014 SCHOLARSHIP EXAMINATION

PRACTICAL SECTION

DEPARTMENT	Computer Science
COURSE TITLE	Year 13 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 may 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. **On The Spot?** (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.

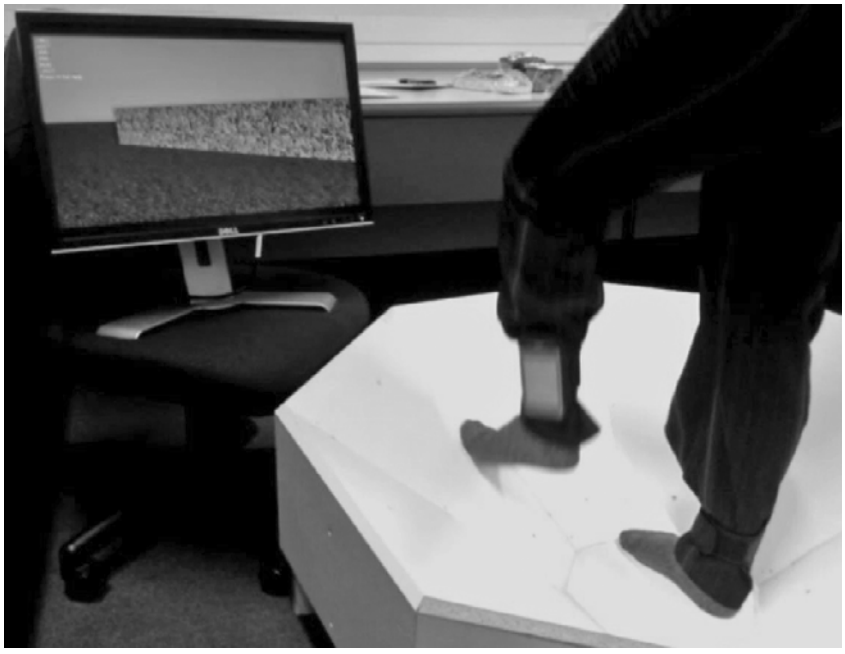
   (i) The accuracy of your results.

   (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.

   While we often think of spreadsheets as tools for financial analysis, they are actually used for many other purposes. In particular, scientists and engineers often use spreadsheets for exploring and analyzing their data. In this question you will be given some data from an experiment and be asked to 'explore' it – find ways of transforming and presenting the data to allow better understanding.

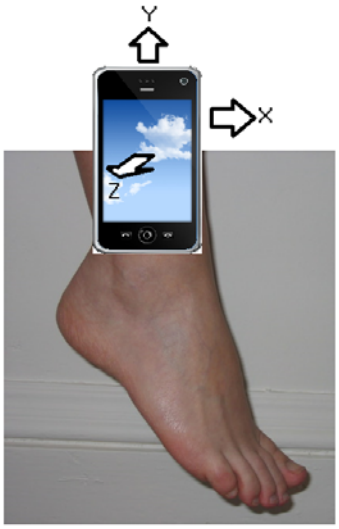
   We were experimenting with the possibility of using cellphones to track the leg motion of a person in a virtual reality environment – the idea being that they might walk 'on the spot' and that the system might interpret this as proper movement in a virtual environment. For this experiment Rachel has cellphones strapped to the sides of each ankle. She is standing in the centre of an octagonal wooden trough with quite steep sides (approx. 45 degrees). If she takes a step in any direction her foot moves up and in the intended direction, then goes down to touch the trough side and slides back down to the centre of the trough. This is a little different from walking on the spot. She can actually move her foot forward, backward or to the side, but will always end up standing in the middle of the trough.



   The cellphones record accelerometer data approximately 100 times per second. You have been supplied with a record of a little over 14 seconds of data from Rachel's right leg. Each row of your data holds one observation with four numbers. The first is the time in

(Question 1 – continued on next page)

milliseconds (1/1000th sec). The other three are the acceleration in X, Y, and Z directions respectively. Here are the first four rows.




```
10.00839758,0.09935937,8.844181,3.5984852
20.01679515,0.10175357,8.826224,3.6367924
30.02519273,0.044292733,8.7951,3.6032736
40.03359031,0.033518825,8.780735,3.7217865
```

The orientation of the axes of the cellphone is approximately as shown in the picture on the right.

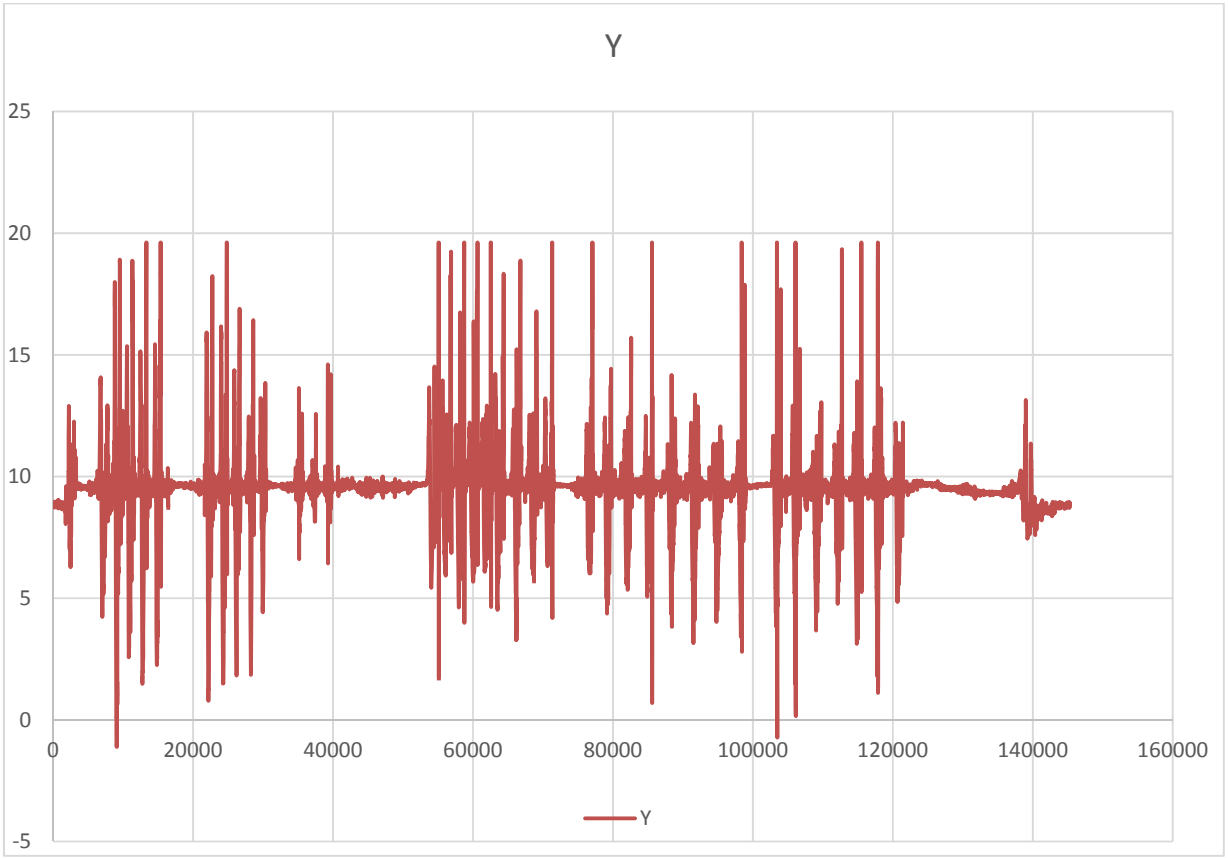
The accelerometer measures acceleration from movement, and also acceleration due to gravity. The acceleration due to gravity is approximately 9.8 (metres/sec^2) mostly in the Y direction.

Step One.

A file called 'rightleg.csv' will be supplied to you. Read the data from this file into a spreadsheet.

Step Two.

Make charts showing acceleration against time. The picture below shows acceleration in the Y dimension. The time axis is in milliseconds (thousandths of a second). Note that the acceleration varies above and below an average value of around 9.8. Note also that the first and last short segments of the graph have lower acceleration. This is because Rachel raised and tilted her ankle to reach the phone for turning on and turning off.



(Question 1 – continued on next page)

Step Three

Your task is to try to make sense of the data. Can you find a way of displaying the data so that each step taken by Rachel is clear? Can you decide in which direction she moved her foot at each step? If this data were being used to control movement in a game, how could you detect a step and decide on its direction?

Some ideas to think about:

Rachel was stepping at an ordinary rate. How long does it take you to make a step? There is a time scale.

Zooming in on part of the data makes it easier to see detail (ie: look at short time periods).

You can integrate acceleration to get speed. If you start with a speed of zero, then in each time interval add the length of the time interval multiplied by the acceleration in that interval, you will obtain approximate speed. Unfortunately you will find that the speeds you get are very large because acceleration values include gravity. Can you find a way of removing gravity from the Y acceleration?

To hand in, write brief notes on your spreadsheet and print out illustrative screen shots to help with the explanation. You can write this on paper, or use a text editor, or a word processor.

CONTINUED

2. **Numbers in Words** (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.

When writing a cheque, the normal practice is to put the amount of money both in digits and in words. When cheques are generated by computer, the software has to be able to do the conversion from numeric amount to words. For example:

Your task is to write a program that generates the English word form for an amount of money in the range $0.00 to $99.99. There are design details to consider in this program. For example: should you convert $1.00 to "one dollar" or to "one dollar and zero cents". Rather than write down all the rules we have provided a range of examples. Your program's output should match that in the sample interaction below. (Underlined text is typed by the user of the program.)

Sample program Interaction

```
Enter an amount: 1.00
one dollar and zero cents
Enter an amount: 1
one dollar and zero cents
Enter an amount: 1.1
one dollar and ten cents
Enter an amount: 1.01
one dollar and one cent
Enter an amount: 1.99
one dollar and ninety nine cents
Enter an amount: 99
ninety nine dollars and zero cents
Enter an amount: 0.99
zero dollars and ninety nine cents
```

3. **Making Tracks** (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.***

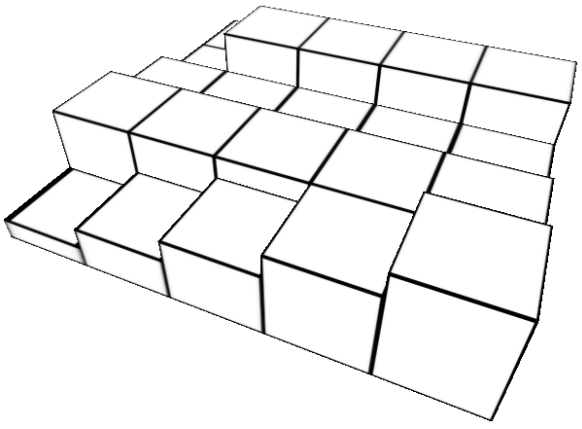
(b) The extent to which your program works and correctly solves the problem.

You may find that the programming language you use makes it difficult to work as shown in the example implementations below. If this is the case, feel free to build your program in a way that suits your circumstances.

Given the current craze for fitness gadgets that measure your movement and exertion levels, it is hardly surprising that people are trying to invent games in the same domain. Your task in this problem is to assist the inventor of a new game called 'Making Tracks' to implement an experimental version of the game, and to offer some advice on the appearance and rules of the game.

The concept of the game is of two players walking over a hilly terrain. Their fitness restricts them to paths that are not too steep (either going up or down). Their goals are to move from a starting point to a finishing point, getting as much exercise as possible. Exercise is measured by distance travelled, with uphill travel giving higher exercise rates than downhill or level travel. The player who gets the most exercise will win.

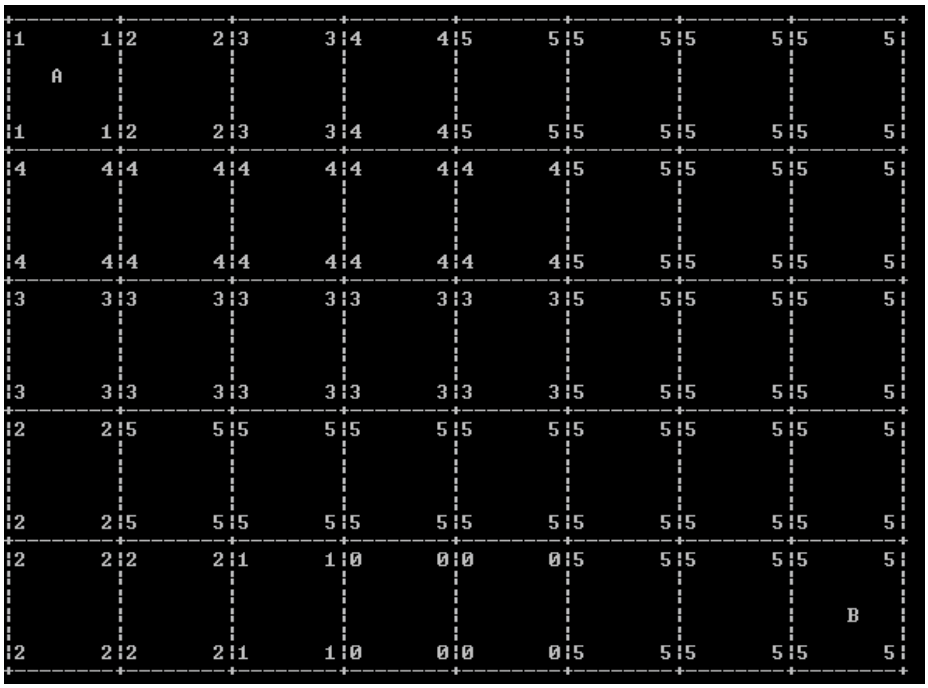
The actual game is played on a rectangular board, divided into squares. There are two players: A and B. A starts at the top left corner of the board and must finish at the bottom right. B works in the opposite direction – from bottom right to top left. At the start of play, the board size (number of rows and number of columns) is decided. Each square of the board is given a height (an integer in the range 0 to 5 inclusive). A physical version of a 5 by 5 square board might look something like this.



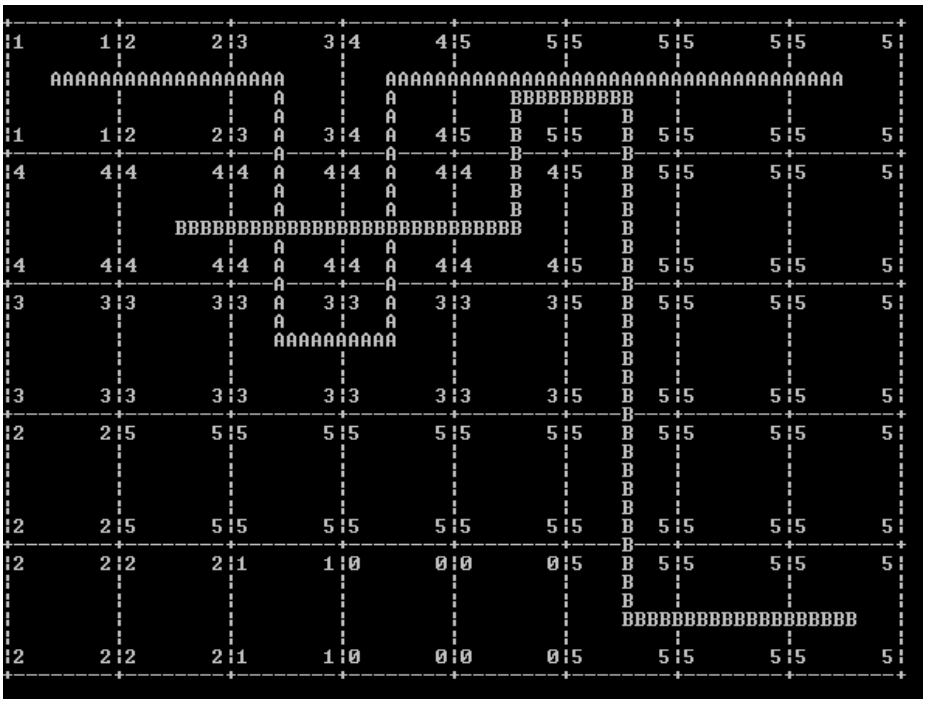
You have been asked to make a text version of the game, displaying the board on the console, and allowing players to enter moves on the keyboard.

(Question 3 – continued on next page)

The starting game state with a 5(row) by 8(column) board might look like as follows. Note that A and B are at their starting squares.



Here is a possible board state after each player has made 11 moves, showing the paths followed by each player.



Question 3- continued on next page

The rules are as follows:

Players take turns to move (except at the end, where one player may continue to move after the other has finished).

At each turn a player may move to an adjacent square (left/right/up/down on the board). Diagonal moves are not permitted. A player may not move off an edge of the board.

A player may only move to a square that has the same height, or whose height differs by no more than one from the height of the player's current square.

A player may not visit any square on the board more than once.

A player may be on the same square as the other player the same time, or may visit squares previously occupied by the other player.

A player scores one point for each move, and an additional point if the move is 'uphill' (from a square of height h to a square of height h+1).

A player loses immediately if they have not reached their finishing square and cannot make a legal move.

The game finishes either when a player has lost, or when both players reach their finish squares. One player may take more moves than the other to reach their finish.

We suggest that you develop your program in the following stages. There is a great deal of programming to be done. Do as much as you can.

Stage 1: Read a game layout (board size and square heights). Details of how you specify the information are up to you.

Stage 2: Display a board with heights for each square.

Stage 3: Allow the user to enter a move. You can choose how a move should be entered.

Stage 4: Check that the move is legal. If not, tell the user why not.

Stage 5: Perform the move and display the updated board, showing the path followed by each player.

Stage 6. Calculate and display a score for each player.

Stage 7. Check for the losing condition.

Stage 8. Announce a winner when both players reach their finish squares.

If you get all that done, then there are two further challenges.

Stage 9. The suggested text layout makes it difficult for players to plan routes, because it is hard to tell which moves are legal (height differences of no more than one). Suggest and implement an improved display which makes it easier to plan routes.

Stage 10. Suggest and implement one change to game play, which in your view would make the game more interesting to play. (This stage requires a brief written explanation of your idea.)