THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

2018 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.
SPECIAL INSTRUCTIONS	Please hand in listings, notes and answers to written questions, and a Pen Drive or DVD with your program/computer work for each question. In addition 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. Candidates may not have access to the internet during the examination.
CALCULATORS PERMITTED	Yes

1. **Anagrams** (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.

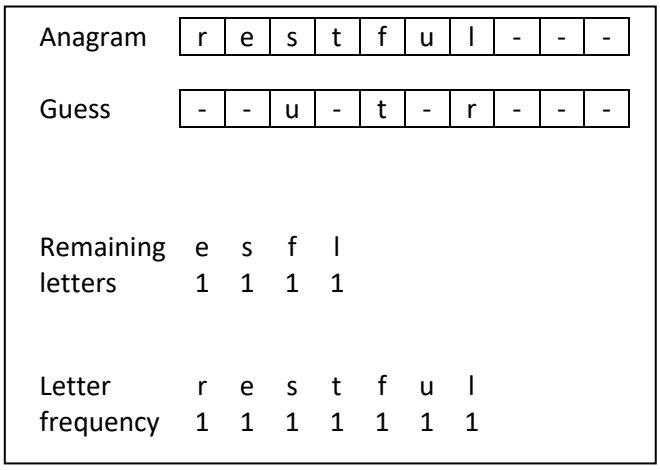
(ii) The skill you show in making use of the capabilities of the spreadsheet.

(iii) The presentation of your results. We have deliberately not provided any instructions concerning layout or formatting. Your goal is to make the spreadsheet easy to use.

This is an unusual way of using a spreadsheet. We normally think of a spreadsheet as suitable mainly for financial calculations. However spreadsheet operations provide quite general capabilities for many programming tasks. In this question you will be guided through developing a spreadsheet 'program' that someone else might use to help them solve 'anagram' style cryptic crossword clues. You are required to solve the problem using spreadsheet operations. You may not use any embedded programming language (such as Visual Basic in Microsoft Excel).

Cryptic Crosswords are a popular puzzle, often found in daily newspapers. To answer this question you do not need to have solved such puzzles yourself. You are being asked to create a tool which puzzle solvers will be able to use to help them with one of the many kinds of clue found in cryptic crossword puzzles. The kind of clue is called an 'anagram'. It has words whose letters must be reorganized to find the answer. For example, the letters of the word 'restful' can be reorganized to form the word 'fluster'. Some anagrams consist of multiple words. For example, 'rail safety' can be reorganized to 'fairy tale', and 'customers' can be reorganized to 'store scum'. In these cases space characters are ignored –anagram puzzles are just concerned with the letters and their ordering.

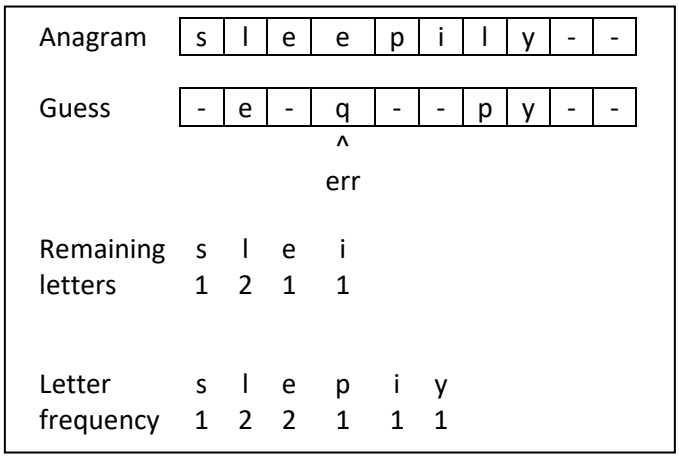
Imagine then the situation of a cryptic crossword solver. From time to time they find themselves with a word or words, and they want to reorganize the letters of those words to see what other words they can form. At the time they start, they may already know the positions of some letters in the solution, and only need to determine the remainder. The screenshot shows part of a sample solution to the problem (column and row lines removed for clarity). The solver can enter letters into the 'guess' and the program keeps track of the letters still available.



(Question 1 – continued next page)

(Question 1 – continued)

It has been designed to support anagrams up to 10 characters long. The top line holds the anagram with hyphens for unused letters. The second line is a guess of some letters, again with hyphens in empty spots. The two lines at the bottom show the letter frequencies for the anagram. The lines in the middle show the letters yet to be used and their frequencies. In the next screen shot the anagram chosen has some repeated letters, making the frequencies more interesting, and also there is a guess with a letter not included, which the program flags as an error.



The following stages should allow you to develop something similar to the example shown.

Stage A

Create a spreadsheet. Make lines to enable a user to enter an anagram, and a guess. Use outline boxes to show where the letters are to be placed. You should decide how large an anagram you will support at this stage. The sample program allowed for 10 characters. A typical small crossword will have clues with a maximum size of 14 characters.

(Question 1 – continued next page)

(Question 1 – continued)

Stage B

The next stage is to 'program' a calculation of the letter frequencies in the anagram. When we program in a spreadsheet, we use cells for all variables. If we are storing an array of data we will use a row or column of cells. Do not be surprised if your spreadsheet starts to get large. The sample used to produce the screenshots above had an area of roughly 100 rows by 100 columns (although less than ¼ of the cells in that area were used).
[Read this entire stage before working, because we present one way of doing the calculation and then show how to reduce the number of cells involved.]

In the following screenshot, the column headings are taken from the anagram input area and the row headings are the letters of the alphabet. Each cell is set to 1 where the column and row headings match using the 'if' function of the spreadsheet. If we sum across each row, we get the frequency with which that letter occurs in the anagram. Only the first 10 rows of the table are shown.

	s	l	e	e	p	i	l	y	-	-	
a	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0
d	0	0	0	0	0	0	0	0	0	0	0
e	0	0	1	1	0	0	0	0	0	0	2
f	0	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	1	0	0	0	0	1
j	0	0	0	0	0	0	0	0	0	0	0

. . . . . more lines from k to z . . . .

Producing just the frequencies of letters used in the anagram, will require removing all the letters with zero frequencies. A trick to reduce the number of zero entries is to just calculate the frequency table for the letters in the anagram. The following table was made by using the anagram letters as the row headings as well as the column headings. This makes a smaller table, but introduces the new issue that some letters occur in more than one row.

(Question 1 – continued next page)

(Question 1 – continued)

	s	l	e	e	p	i	l	y	-	-	
s	1	0	0	0	0	0	0	0	0	0	1
l	0	1	0	0	0	0	1	0	0	0	2
e	0	0	1	1	0	0	0	0	0	0	2
e	0	0	1	1	0	0	0	0	0	0	2
p	0	0	0	0	1	0	0	0	0	0	1
i	0	0	0	0	0	1	0	0	0	0	1
l	0	1	0	0	0	0	1	0	0	0	2
y	0	0	0	0	0	0	0	1	0	0	1
-	0	0	0	0	0	0	0	0	1	1	2
-	0	0	0	0	0	0	0	0	1	1	2

Stage C

The next stage, assuming that you used the trick presented in Stage B, is to correct the letter frequencies where letters occur more than once. The full count should be associated with the first occurrence of a repeated letter. The other occurrences should have count zero.

You will need to add extra cells (even whole copies of the table) to your spreadsheet in order to perform this calculation. Remember, when a calculation involves several steps, new cells must be introduced for each step.

The new frequencies should be as shown on the right.

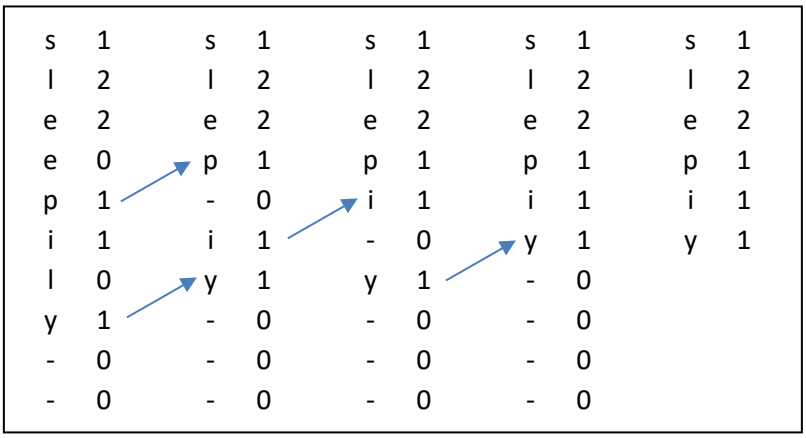
s	1
l	2
e	2
e	0
p	1
i	1
l	0
y	1
-	2
-	0

(Question 1 – continued next page)

(Question 1 – continued)

Stage D

To produce the neat list of letter frequencies shown in the sample program output, all letters with zero frequencies, and the '-' character entries must be removed. The method used in the sample program based on the idea of a 'shifter' block of cells. A shifter block identifies rows with non zero frequency directly below lines with zero frequency, and shifts those lines up one position. In the screen shot below the columns necessary to perform this action have been removed, and only the results are shown. A final column blanks out the lines with zero frequency. For the example only 3 shifters are shown. Extra shifters do no harm, so you should have enough to deal with largest number of shifts that could occur.



(Question 1 – continued)

Stage E

The sample program also showed letter frequencies for letters not yet used. Here are two ideas towards developing that feature of the program.

1. You should now have a group of spreadsheet cells that can calculate letter frequencies for the anagram. If you copy that whole group of cells and paste it to a clear area of your spreadsheet, it should be possible to supply the guess characters as its input and therefore produce letter frequencies for the letters in use.
2. If you subtract frequencies of letters in use from the frequencies of those letters in the anagram, you should end up with the unused letter frequencies.

Stage F (Final)

The sample program also featured error detection – in that it noticed if a letter that did not occur in the anagram was used. Add that capability to your program.

2. **Change** (Careful and Accurate Programming)

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

(a) Completeness and accuracy of the program. It may be that this problem statement does not state exactly what the program should do under all circumstances. If you find a situation of that nature, choose a solution and write down, either on paper or in the comments of your program what the difficulty was and how you chose to resolve it.

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

New Zealand coins come in the following values: 10c, 20c, 50c, $1 and $2. Notes come in the values: $5, $10, $20, $50 and $100. People mostly pay for goods in shops using some form of card, but it is still possible to pay in cash. A person paying in cash does not usually present the exact amount of money required. Instead they give a larger value in cash and the shop assistant gives change. Change should be given in as few notes and coins as possible. For example: if a customer provides $10 in payment for an item that costs $7.40, the assistant must give $2:60 in change. There are many ways that amount could be made up – for example two $1 coins and three 20c coins. The correct solution, involving only 3 coins is one $2, one 50c and one 10c.

An additional problem is that prices are often values that cannot be exactly represented in notes and coins, and lead to situations where it is not possible to give correct change. For example: a customer giving a $5 note for goods costing $4.99 cannot be given change because there is no 1c coin. In that case the shop assistant must round off the price. The method most widely used in New Zealand is called Swedish rounding. Here are the rules:

- Prices are rounded down to the nearest multiple of 5 cents for sales ending in: 1¢ & 2¢ (rounded to 0¢); and, 6¢ & 7¢ (rounded to 5¢).
- Prices are rounded up to the nearest multiple of 5 cents for sales ending in: 3¢ & 4¢ (round to 5¢); and, 8¢ & 9¢ (round to 10¢).
- Values ending in 0¢ or 5¢ remain unchanged.

Your task is to write a program which reads a price and an amount of money provided by a customer. It should display the best way of providing change.

A sample interaction with your program is given on the next page. User input is underlined.

(Question 2 – continued next page)

(Question 2 – continued)

```
Welcome to the change program

Enter price of goods: $7.40
Enter amount paid:    $10
Rounded price:        $7.40
Change:               $2.60
        Notes: None
        Coins: $2, 50c, 10c

Enter price of goods: $7.60
Enter amount paid:    $20
Rounded price:        $7.60
Change:               $12.40
        Notes: $10
        Coins: $2, 20c * 2

Enter price of goods: $10
Enter amount paid:    $10
Rounded price:        $10.00
Change:               $0.00
        No change is needed
```

3. **Bubbles** (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. You may include comments in your program, or write a description of your program to hand in.***

(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 produce output as shown in the example implementation steps below. If this is the case, feel free to build your program in a way that suits your circumstances.

The game Tetris was invented and first implemented by Alexey Pajitnov in June 1984. His implementation was for a keyboard and text console. This question asks you to implement a new game, called 'Bubbles', in a similar style.

The game is about diamond mining. A diamond is embedded in rock in a mine shaft and the player must blast the rock away to expose and release the diamond. The mine is full of a dangerous gas which explodes when it is trapped in bubbles. It is too dangerous to enter the mine. Instead miners throw shaped pieces of rock into the shaft. The rock pieces stick to the mine surface. Careful placement of pieces can create enclosed bubbles of mine gas that explode, destroying themselves and parts of the mine rock. Repeated explosions will eventually release the diamond.

We will present the problem in stages for you to program. The stages are interleaved with explanation of aspects of the game. We suggest that you build your program in the order given. This will make it likely that you have parts working at the end, even if you don't have time to complete the whole program. However, we also strongly suggest that you read the whole problem statement before starting to program. We also suggest that you save working versions of your program at each stage.

Stage A: Draw the mine shaft.

The lines of '-' characters top and bottom, and the column of '|' at the right form the boundaries of the mine shaft. There are 7 space characters to the left of the shaft top and bottom walls. This creates a launching space for rocks being thrown into the mine. The '@' characters are rock filling the end of the shaft. The 'D' character is the diamond.

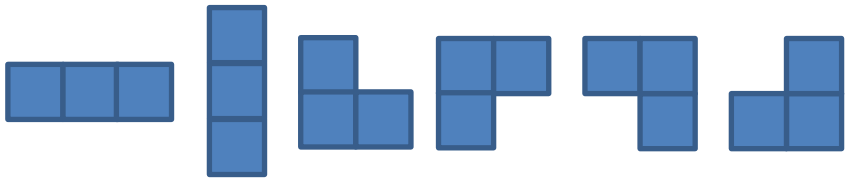
```
       --------------------------------------------------
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@D@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
       --------------------------------------------------
```

(Question 3 – continued next page)

(Question 3 – continued)

Stage B: Put a rock shapes in the launch area

The rock shapes are 'triominoes' – shapes made from three cubes. The two possible shapes and their rotations are:



Drawn in character graphics, using '@' because they are made of rock, they look like this:

```
          @
@@@       @       @       @@      @@       @
          @       @@      @         @      @@
```

When the games starts, and at the start of each move, a random character is placed in the launching area. It is placed in its first rotation. I.e. the bar shape is horizontal and the angled shape has its gap at upper right. Here is the game with the bar in launch position.

```
     ------------------------------------------------
                                              @@@@@@@@@@|
                                              @@@@@@@@@@|
                                              @@@@@@@@@@|
@@@                                           @@@@@@@@@D@|
                                              @@@@@@@@@@|
                                              @@@@@@@@@@|
                                              @@@@@@@@@@|
     ------------------------------------------------
```

Stage C: Prompt for a command

Commands are typed. A single command line describes a launch. It is a sequence of the letters 'u' for up (move the launch position up by one line), 'd' for down (move the launch position down by one line) and 'r' for rotate (rotate the shape clockwise 90 degrees). In this screen shot the player has typed 'uur'. There is no time limit – players can ponder moves for as long as they wish. Nothing happens until they hit the 'Enter' key.

```
     ------------------------------------------------
                                              @@@@@@@@@@|
                                              @@@@@@@@@@|
                                              @@@@@@@@@@|
@@@                                           @@@@@@@@@D@|
                                              @@@@@@@@@@|
                                              @@@@@@@@@@|
                                              @@@@@@@@@@|
     ------------------------------------------------

Move (r/u/d): uur
```

(Question 3 – continued next page)

(Question 3 – continued)

Stage D: Enact commands

The letters of the command are enacted in order. If an up/down command cannot be performed because it would push the piece up or down so that it would not be inside the lines of the mine shaft, or if a rotation cannot be performed for the same reason, then that command is ignored and the next is considered. Here is the game before and after a command line is executed. [*You may find it best to add this functionality in small stages. Try individual commands and check that the results are as you would like.*]

```
        --------------------------------------------------
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
    @@@                                         @@@@@@@@@D|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
        --------------------------------------------------

Move (r/u/d): ruuuuu


        --------------------------------------------------
     @                                          @@@@@@@@@@|
     @                                          @@@@@@@@@@|
     @                                          @@@@@@@@@@|
                                                @@@@@@@@@D|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
        --------------------------------------------------
```

(Question 3 – continued next page)

(Question 3 – continued)

Stage D: Throw the shape

There is no special command for throwing. As soon as the commands entered by a player have been enacted, the shape is thrown automatically. It moves right until it is blocked by hitting rock or the end of the shaft. Note that a player may just hit return to throw the shape without movement or rotation. Here is a sequence of two throws.

```
       --------------------------------------------------
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
   @@@                                          @@@@@@@@@D@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
       --------------------------------------------------

Move (r/u/d):

       --------------------------------------------------
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
   @@@                                       @@@@@@@@@@@D@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
       --------------------------------------------------

Move (r/u/d): ur

       --------------------------------------------------
                                                @@@@@@@@@@|
                                             @    @@@@@@@@@@|
                                             @    @@@@@@@@@@|
   @@@                                       @@@@@@@@@@@@@D@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
                                                @@@@@@@@@@|
       --------------------------------------------------
```

(Question 3 – continued next page)

(Question 3 – continued)

Stage E: Detect bubbles

After a number of throws, enclosed areas may be formed. One more throw in the example of the last page might look like this. Note the gas bubble enclosed by rock. There is one special rule to add: a bubble must have more than one connected cell – a single gap enclosed is not a valid bubble. Sides and the end of the mine shaft can be used as part of the enclosure of a bubble. Gas cannot escape via diagonals, so the top left corner of the bubble in the example is safely enclosed.

```
Move (r/u/d): uuuuuuu

       ------------------------------------------------
                                          @@@@@@@@@@@@@|
                                        @    @@@@@@@@@@@|
                                        @    @@@@@@@@@@@|
                                        @@@@@@@@@@@@@D@|
                                            @@@@@@@@@@@|
                                            @@@@@@@@@@@|
                                            @@@@@@@@@@@|
       ------------------------------------------------
```

Stage F: Explode bubbles

Explosion removes the boundary of a bubble. The bubble of the last screen shot removes rock as shown by '*' characters in the following screen shot. Note that diagonals are included in the destruction. Some of the original rock of the mine is removed in this example. Isolated rock fragments, unconnected to the main area of rock may remain after an explosion – they are just left where they are.

```
       ------------------------------------------------
                                           ****@@@@@@@@@|
                                         *    *@@@@@@@@@|
                                         *    *@@@@@@@@@|
                                         *****@@@@@@@@D@|
                                            @@@@@@@@@@@|
                                            @@@@@@@@@@@|
                                            @@@@@@@@@@@|
       ------------------------------------------------
```

(Question 3 – continued next page)

(Question 3 – continued)

Stage G: Winning and losing.

A player loses the game if rock has built up to the point that a new shape cannot be put in the launch position. A player wins when an explosion removes the diamond. Here is a game configuration just before the winning play and after the play, showing a giant bubble just before it explodes.

```
       ------------------------------------------------------
                                                          @|
                                                 @        @|
    @                                           @@@@@@@@@@@@|
    @@                                        @             D|
                                              @@@@          @|
                                                @     @ @   @|
                                                @  @@@@@@@@@@|
       ------------------------------------------------------

Move (r/u/d): u


       ------------------------------------------------------
                                                          @|
                                               @    @     @|
                                               @@@@@@@@@@@@@|
                                              @             D|
                                              @@@@          @|
                                                @     @ @   @|
                                                @  @@@@@@@@@@|
       ------------------------------------------------------

Congratulations, YOU WON!!!!

Play again ('YES'):-
```