THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

2012 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 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. **Who's The Best?** (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.

Every four years the Olympic Games provide a focus for national pride and an opportunity to compare the countries of the world. However, this is not as easy a comparison to make as we might think? Is it more meritorious for a small country or a large country to win a medal? Are we more impressed by a poor country winning a medal than a rich country? The Guardian(an English newspaper) published an analysis of the comparison problem in July 2012. Here are a few extracts from their article.

How do you measure a team's performance in the Olympics? The traditional way is to just count up the number of medals won. And the result? The biggest countries always come top: the Olympic 'superpowers' of the US, China, Russia, UK, Australia and Germany. But what if the totals took account of factors that must have an influence, such as the size of a country's population or its economic power, or compared it to the size of the athletic team in London?

Take the 2008 results. The Bahamas had a population of approximately 334,000 in 2008, whereas the USA had 304,000,000 - almost 1,000 times larger. And yet the Bahamas won two medals, whereas the US 110 – 55 times as many. Taking population into account, "It no longer seems obvious that the US should rank higher than the Bahamas.

GDP [Gross Domestic Product – the total value of goods and services bought and sold per year in a country – a measure of economic power]*, is another obvious one to re-rank on, particularly when you consider how expensive sport equipment and training is. Moreover, since GDP also grows with population size, it implicitly also takes into account population size.*

Team size is also a factor - and one which may be a better indicator than either GDP or population as it has already taken those into account by team selection.

Your task is to analyse some statistics we provide in order to investigate the relationship between Medal Count, Population, GDP and Team Size

Step One

You have been provided with two files containing useful data. As commonly occurs in data analysis some reorganization of the data may be necessary to get it into a format suitable for analysis. The first step in answering this question is to get the data from both files into a single spreadsheet, with one line(row) for each country.

The first file (medals.csv) has one line for each country that won at least one medal. Each line has the name of the country with its three letter Olympic Committee name code in parentheses; the number of gold medals won, the number of silver medals and the number of bronzes. The items on each line are separated by commas – hence the file being a .csv or comma separated value file. The first few lines look like this.

(Question 1 - continued on next page)

```
United States (USA),46,29,29
China (CHN),38,27,23
Great Britain (GBR),29,17,19
Russia (RUS),24,26,32
```

The second file has country statistics. There is one line for each country (including those that didn't win any medals). Again, separated by commas, each line has: the country name; a three letter country code; the GDP (in $); the population; the number of athletes in its Olympic team; and a second country code. One of the country codes is the Olympic Committee code; the other is the International Standard Code. One will match the codes used in the medal table, one may not.

```
Afghanistan,AFG,20343461030,34385000,6,AFG
Albania,ALB,12959563902,3205000,11,ALB
Algeria,DZA,188681000000,35468000,39,ALG
```

Step Two

Produce three graphs (charts). The first should show the relationship between the total number of medals won and the GDP of the countries. The second should show the relationship between total medals and population. The third should show the relationship between total medals and team size. Your challenge is to display this data in a way that allows someone to see any relationship – by careful choice of graph size and scales.

Step Three

Consider the relationship between total medals and GDP. By looking at your graph, or otherwise, estimate the number of medals you would expect a country to win per billion dollars of GDP . Produce another spreadsheet or graph which clearly shows those countries that are winning more medals and those winning fewer, than expected on the basis of their GDP.

Do the same for population and team size.

Step Four

Having looked at your data, a person might wonder "Which countries are doing better than expected and which are doing worse, on all three measures". Make a display that helps answer this question?

2. **How Many Letters** (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.

Charles Babbage and Ada Lovelace are widely recognized as the parents of modern computing. Babbage designed a general purpose programming engine and Lovelace worked out how to program it. A little known fact about Babbage however, is that he was also interested in cryptography. In 1854 he cracked the **Vigenère Cypher,** an encryption system considered unbreakable since its invention in 1553, although this fact was not made public until the 1970's.

Given an enciphered message to crack, the first thing we usually do is to count the number of times each letter occurs. Then it is possible to make guesses based on the frequency of letters in English (or whatever language you are working with). In this question, given a string, you are required to count the number of times each letter occurs.

- Only count letters of the alphabet – ignore spaces, punctuation, etc
- Treat upper and lower case letters as being the same – 'A' and 'a' are the same
- Report only counts of letters that actually occur in the string

An interaction with your program might look like this (underlined text is entered by the user of the program:

```
Enter text >> Alphabet

   A occurs 2 times
   B occurs 1 time
   E occurs 1 time
   H occurs 1 time
   L occurs 1 time
   P occurs 1 time
   T occurs 1 time

Enter text >> It's time for more COMPLEXITY

   C occurs 1 time
   E occurs 3 times
   F occurs 1 time
   I occurs 3 times
   L occurs 1 time
   M occurs 3 times
   O occurs 3 times
   P occurs 1 time
   R occurs 2 times
   S occurs 1 time
   T occurs 3 times
   X occurs 1 time
   Y occurs 1 time
```

CONTINUED

3. **Image Cleaning** (Problem Solving and Programming)

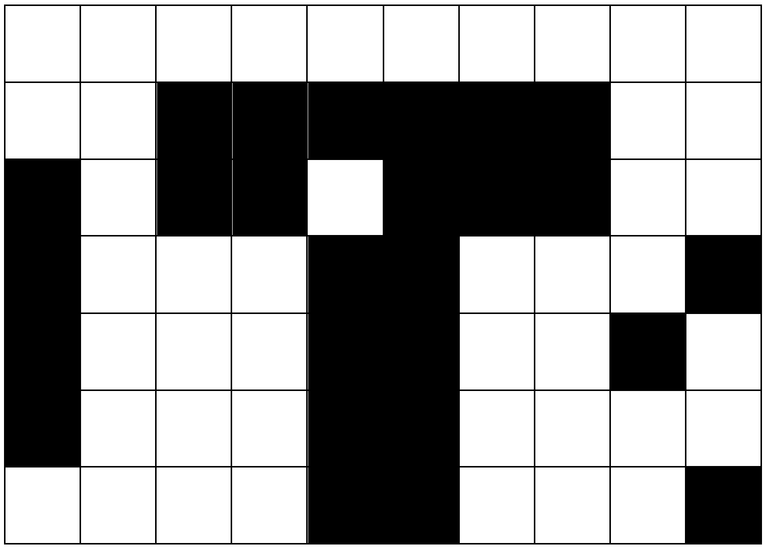
Your programming work in this question will be assessed on two criteria:

(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 way of working.

Nowadays most documents (books, reports, legal documents, exam papers, etc.) are produced on computer using a word processor. If we want to, we can add them to databases or put them on the web, gaining all the advantages of digital media – being able to search a collection of documents, produce modified versions, etc. Unfortunately there are a large number of important old documents in libraries and other collections, which were produced before the digital era. A useful technology for capturing this old data and getting it into digital form is scanning and character recognition. Character recognition is not easy to program. Images of documents (particularly old documents that may have suffered damage of some sort) tend to be untidy. The example below shows a highly zoomed-in part of a document scan, with a letter 'T'. Each box in the image is one pixel as returned by the scanner. It is likely that the white pixel in the 'T' is a scanning error as are the two isolated black pixels to the lower right. The bar of four black pixels to the left, however, is probably part of a letter to the left of the 'T'. The first step in character recognition program is usually to 'clean' the image – to try to get rid of isolated incorrect pixels, or small groups of incorrect pixels. Your task is to write a program to clean an image.



(Question 3 - continued on next page)

TURN OVER

Stage 1: The first step is to be able to read an image into your program. You are reasonably free to do this however you like, but your program must have a way to enter new images. We will assume for this program that the images are always small (no more than 15 by 15 pixels). A way of doing this at the command line is to have an interaction something like this (underlined text entered by the program user):

```
How many rows:    7
How many columns: 10
Enter numbers for black pixels
  Row 0:
  Row 1:  2 3 4 5 6 7
  Row 2:  0 2 3 5 6 7

  .  .  .
```

Stage 2: Write instructions to display an image. Using the console, it is possible to produce a clear display. Here is an example. The exact format is up to you.

```
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |###|###|###|###|###|###|   |   |
|   |   |###|###|###|###|###|###|   |   |
+---+---+---+---+---+---+---+---+---+---+
|###|   |###|###|   |###|###|###|   |   |
|###|   |###|###|   |###|###|###|   |   |
+---+---+---+---+---+---+---+---+---+---+
|###|   |   |   |###|###|   |   |   |###|
|###|   |   |   |###|###|   |   |   |###|
+---+---+---+---+---+---+---+---+---+---+
|###|   |   |   |###|###|   |   |###|   |
|###|   |   |   |###|###|   |   |###|   |
+---+---+---+---+---+---+---+---+---+---+
|###|   |   |   |###|###|   |   |   |   |
|###|   |   |   |###|###|   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |###|###|   |   |   |###|
|   |   |   |   |###|###|   |   |   |###|
+---+---+---+---+---+---+---+---+---+---+
```

(Question 3 - continued on next page)

Stage 3: Write instructions to clean the image. The first part of cleaning is to get rid of isolated black pixels and small groups of black pixels. The ideas are as follows. You may like to find solutions for parts of the problem separately, or you may find a general solution that covers them all.

- A pixel has 8 neighbours – horizontally, vertically and diagonally (except edge and corner pixels which have fewer)
- A black pixel is said to be isolated if all of its neighbours are white.
- Isolated black pixels should be changed to white.
- Two black pixels are said to be connected if they are neighbours.
- Two connected black pixels are an isolated group if neither has any other black neighbour.
- Isolated groups of two black pixels should be changed to white.

Extend your program following these ideas, to remove isolated black pixels or isolated groups of two black pixels. Modify your display to show the result, indicating the pixels that have been turned white in some way.

```
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |###|###|###|###|###|###|   |   |
|   |   |###|###|###|###|###|###|   |   |
+---+---+---+---+---+---+---+---+---+---+
|###|   |###|###|   |###|###|###|   |   |
|###|   |###|###|   |###|###|###|   |   |
+---+---+---+---+---+---+---+---+---+---+
|###|   |   |   |###|###|   |   |   |...|
|###|   |   |   |###|###|   |   |   |...|
+---+---+---+---+---+---+---+---+---+---+
|###|   |   |   |###|###|   |   |...|   |
|###|   |   |   |###|###|   |   |...|   |
+---+---+---+---+---+---+---+---+---+---+
|###|   |   |   |###|###|   |   |   |   |
|###|   |   |   |###|###|   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |###|###|   |   |   |...|
|   |   |   |   |###|###|   |   |   |...|
+---+---+---+---+---+---+---+---+---+---+
```

Stage 4: We can apply similar rules to repair white patches in areas of black. Extend your program to do this. Does it work? Why? If there is a problem, suggest a solution.