Marking scheme:

For each question:

0.4/2.0 Marks will be given to students who have submitted the program on time.

0.8/2.0 Marks will be given to students who wrote the program that can be compiled without errors

1.6/2.0 Marks will be given to students who wrote the program that meet all the requirements of the questions.

Additional 0.2 marks will be given to students who demonstrate good programming habit and style. For example, comments, meaningful variable name given, etc.

(P.S. A good coding habit is essential for a programmer when working with collaborators or manager in the industry. Please pay attention to this.)

Additional 0.2 marks will be given to student who demonstrate superior programming skills and knowledge.

For example, the minimization of the use of variables, memories and elapsed times by good programming skills, etc

Reducing program complexity or define the functions that allows the program to be run efficiently, etc

Additional information for the questions:

Question 1

You must complete the labelgen.h and labelgen.cpp so that the following codes can run:

The TestLabelGenerator.cpp will looks like this:

```
/*
 * File: TestLabelGenerator.cpp
 * ----------------------------
 * This program tests the labelgen abstraction by using it to
generate
 * a sequence of labels.
 */

#include <iostream>
#include "labelgen.h"
using namespace std;

/* Main program */

int main() {
   LabelGenerator figureNumbers("Figure ", 1);
   LabelGenerator pointNumbers("P", 0);
   cout << "Figure numbers: ";
   for (int i = 0; i < 3; i++) {
      if (i > 0) cout << ", ";
      cout << figureNumbers.nextLabel();
   }
   cout << endl << "Point numbers:  ";
   for (int i = 0; i < 5; i++) {
      if (i > 0) cout << ", ";
      cout << pointNumbers.nextLabel();
   }
   cout << endl << "More figures:   ";
   for (int i = 0; i < 3; i++) {
      if (i > 0) cout << ", ";
      cout << figureNumbers.nextLabel();
   }
   cout << endl;
   return 0;
}
```

Question 2

A PegSolitaire class should be created.

Marks will not be given if Recursion techniques are not used in the question.

The result should display a sequence of jumps:

Example: b4->d4 c2->c4 a3->c3

Question 3

Functions should be included:

```
void printHeader();
void countComparisons(int nElements, int nTrials);
void searchTrial(int nElements, int & nLinear, int & nBinary) {
   Vector<int> vec;
   for (int i = 0; i < nElements; i++) {
      vec.add(randomInteger(1, nElements));
   }

   int key = randomInteger(1, nElements);
   linearSearch(key, vec, nLinear);
   binarySearch(key, vec, nBinary);

}
int linearSearch(int key, Vector<int> & vec, int & ncmp);
int binarySearch(int key, Vector<int> vec, int & ncmp);
int binarySearch(int key, Vector<int> vec, int p1, int p2, int &
ncmp);

nElements is the number of elements in the array
nTrials is the number of independent trials made for that element
count.
ncmp is the counter
key is the value you want to find in the integer list
vec is the list of integer
```

Question 4:

The following code will be used to test your program:

```
const unsigned int numberOfinteger=30;

int main() {

    int array[numberOfinteger];

    for (int i=0;i<numberOfinteger;i++)

    {

         array[i] = randomInteger(-1000,1000);

    }

   sort(array, numberOfinteger);

   printArray(array, numberOfinteger);

   return 0;

}
```

Question 5:

File to be submitted: "mystring.h"

Your program should demonstrate your knowledge in Dynamic Memory.