UNIVERSITY OF SWAZILAND

FACULTY OF SCIENCE & ENGINEERING

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

PROGRAMMING TECHNIQUES II

COURSE CODE – EE272

MAIN EXAMINATION

MAY 2013

DURATION OF THE EXAMINATION - 3 HOURS

INSTRUCTIONS TO CANDIDATES

(a) There are **FIVE** questions in this paper. Answer questions **1** & **2**, and any other **TWO** questions.

(b) Each question carries equal marks.

(c) Show all your steps clearly in any calculations.

(d) State clearly any assumptions made.

(e) Start each new question on a fresh page.

Question 1

(a) What is the difference between an object and a class? [5]

(b) Using an example, explain what is a *access specifier*? [4]

(c) What are constructors used for and how are they defined? [4]

(d) The definition of classes should promote encapsulation. Explain. [2]

(e) Explain the difference between an actual parameter and a formal parameter? [3]

(f) How are overloaded methods distinguished from each other? [3]

(g) What is the difference between a static class member and an instance class member? [2]

(h) Using an example, explain the difference between an *object pointer* and an *object reference*. [2]

Question 2

(a) How does inheritance promote software reuse? [1]

(b) Why would a derived class override one or more of the methods in its base class? [1]

(c) What is the role of an abstract class? [2]

(d) What is polymorphism? [3]

(e) How does inheritance support polymorphism? [3]

(f) How is overriding related to polymorphism? [4]

(g) Discuss how polymorphism makes software systems extensible and maintainable? [5]

(h) In what ways might a thrown exception be handled? [3]

(i) What is object persistence and how is it accomplished in C++? [3]

Question 3

Analyse the following programs and determine their outputs. Show all working.

(a)

```
#include <iostream>

using namespace std;
using namespace System;

int main(void){
      int i, j, c = 9, m, k;
      for (i = 1; i <= 5; i++) {
            for (k = 1; k <= c; k++) {
                  cout << " ";
            }
            for (j = 1; j <= i; j++) {
                  cout << j;
            }
            for (m = j - 2; m > 0; m--) {
                  cout << m;
            }
            cout << endl;
            c = c - 2;
      }
      Console::ReadKey();
    return 0;
}
```

[9]

(b)

```
#include <iostream>

using namespace std;

int main(){

      char prnt = '*';
      int i, j, k, s, nos = -1;

      for (i = 5; i >= 1; i--) {
            for (j = 1; j <= i; j++) {
                  cout << " ";
            }
            for (s = nos; s >= 1; s--) {
                  cout << prnt;
            }
            for (k = 1; k <= i; k++) {
                  if (i == 5 && k == 5) {
                  continue;
                  }
                  cout << " ";
            }
            nos = nos + 2;
            cout << endl;
      }
      nos = 5;
      for (i = 2; i <= 5; i++) {
            for (j = 1; j <= i; j++) {
                  cout << prnt;
```

```
            }
            for (s = nos; s >= 1; s--) {
                  cout << " ";
            }
            for (k = 1; k <= i; k++) {
                  if (i == 5 && k == 5) {
                        break;
                  }
                  cout << prnt;
            }
            nos = nos - 2;
            cout << endl;
      }
    return 0;
}
```

[16]

Question 4

Create a C++ class called `Rational` for performing arithmetic with fractions. Use integer variables to represent the `private` data of the class – the `numerator` and the `denominator`. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializers are provided and should store the fraction in reduced form. For example, the fraction:

$$\frac{2}{4}$$

would be stored in the object as 1 in the `numerator` and 2 in the `denominator`. Provide `public` member functions that perform each of the following tasks:

(a) Input and Output member functions:

(i) ***Write***: serializes a Rational object and stores it in binary file. [4]

(ii) ***Read***: reads a Rational object from a binary file. [4]

(b) Arithmetic member functions:

(i) ***Add***: to calculate the sum of two `Rational` objects. The result should be stored in reduced form. This function overloads the "+" operator. [6]

(ii) ***Subtract***: to calculate the difference between two `Rational` objects. The result should be stored in reduced form. This function overloads the "-" operator. [6]

Write a C++ program to test your class. Your program should provide the user with an appropriate menu to exercise each of the public member functions in class `Rational`. Feel free to add any other private functions you may deem necessary for the `Rational` class and your programs. [5]

Question 5

A college administrator requires a program that reads in test scores and applies two different curves to them. The program should contain a base class *ScoreBank* with two private data members: an integer array for the scores and a float for the average. The maximum number of scores is 10. The class should contain a method *EnterScores*

which asks the user how many test scores are needed and reads in the scores. The class should also contain a method *CalcAverage* which stores the average of the entered scores in the private float data member. Scorebank should also have an *Output* function that prints a sorted list of test scores to the screen as well as the average.

Derive from *ScoreBank* a class called *Curve1* which contains a method *Curve.* This curve sets the average score to 75, finds out how far away from 75 the actual average is, and then add this value to each test score. Overload the *Output* method to print, sorted, the original scores and the curved scores as well as the original and new average.

Derive from ScoreBank a class called *Curve2* which contains a method *Curve.* This curve sets the highest score to 100. The method then finds out how is the highest score from 100 and then adds the difference to each score. Overload the *Output* function to print the original scores, the new scores, and the averages for both sets.

(i) Write the interfaces of each of the three classes. [6]

(ii) Write the implementations of the classes. [19]

END OF PAPER