UNIVERSITY OF SWAZILAND

FACULTY OF SCIENCE AND ENGINEERING

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

MAIN EXAMINATION 2016

Title of Paper:	Programming Techniques II
Course Number:	EE272
Time Allowed:	3 hrs

Instructions:

1. There are **five** (5) questions in this paper. Answer question 1 and any other **three** (3) questions.
2. Each question carries 25 marks.
3. State clearly any assumptions made.
4. Start each new question on a fresh page

THIS PAPER SHOULD NOT BE OPENED UNTIL PERMISSION HAS BEEN GIVEN BY THE INVIGILATOR.

This paper contains nine (7) pages including this page.

<u>Question 1</u>

a) What is the difference between an object and a class? [3]

b) Using an example, explain what is an *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) Using examples, discuss four ways by which class templates and inheritance are related. [5]

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

g) Using an example, explain the difference between an *object pointer* and an *object reference.* [4]

Question 2

a)

(i) How is it that polymorphism enables programming "in the general" rather than "in the specific"? [1]

(ii) Discuss two advantages of programming "in the general". [3]

b)

(i) Discuss two problems of programming with the switch logic. [2]

(ii) Using an example, explain how polymorphism can be an effective alternative to switch logic. [2]

c) Information hiding is one of the key features that distinguish object-oriented programming from structured programming. Using an example, explain the rationale of information hiding and how it relates to the following object-oriented programming concepts: *abstraction, coupling,* and *cohesion.* [4]

d) Discuss the ways in which inheritance promotes software reuse, saves time during program development and helps prevent errors. [4]

e) Describe the ways by which a derived class may inherit from a base class. [5]

f) Using an example, explain the relationship between function templates and function overloading? [4]

Question 3

Carefully analyse the following programs and determine their outputs. Show all your working.

(a) **[15]**

```
#include <iostream>

using namespace std;

int main(void) {

int m, n;

int x = 1, y = 10;

    for (m=x; m<=y; m++){

        for (n=x; n<=y; n++) {

                if ((m<=x+1)||(m>=y-1)) {

                       cout<<"* ";

                } else {

                  ((y-n+1>=m-1)&&(y-n+1<=(m+1))) ? cout<<"* ":cout<<" ";

                }

        }

        cout<< endl;

    }

    return 0;

}
```

(b) **[10]**

Class Interface

```
#pragma once

class DemoProg1 {
      public:
        DemoProg1 (void);
        ~DemoProg1(void);
} ;
```

Class Implementation

```
#include "DemoProg1.h"

#include <iostream>

using namespace std;

DemoProg1::DemoProg1(void){

     int k, num=30;

     k = (num>5 ? (num <=10 ? 100 :200): 500);

     cout << k <<" "<< num << endl;

}

DemoProg1::~DemoProg1(void){

}

int main(void) {

     DemoProg1 dp1;

     return(0);

     }
```

Question 4

(Rational Class)
Create a class called `Rational` for performing arithmetic with fractions. Write a program to test your class.
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's 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) Adding two `Rational` numbers. The result should be stored in reduced form. [5]

b) Subtracting two `Rational` numbers. The result should be stored in reduced form. [5]

c) Multiplying two `Rational` numbers. The result should be stored in reduced form. [4]

d) Dividing two `Rational` numbers. The result should be stored in reduced form. [4]

e) Printing `Rational` numbers in the form `a/b`, where `a` is the numerator and `b` is the denominator. [4]

f) Printing `Rational` numbers in floating-point format. [3]

Question 5

Package-delivery services, such as FedEx, DHL, and UPS, offer a number of different shipping options, each with specific costs associated.

Create an inheritance hierarchy in the form of a class diagram to represent the various types of packages. Use `Package` as the base class of the hierarchy, then include classes `TwoDayPackage` and `Overnight` that derive from `Package`. Base class `Package` should include data members representing the *name, address, city,* and *region* for both the sender and the recipient of the package, in addition to data members that store the *weight* (in kilograms) and *cost per kilogram* to ship the package. Package's constructor should initialise these data members. Ensure that the weight and cost per kilogram contain positive values.

Package should provide a public member function `calculateCost` that returns a double indicating the cost associated with shipping the package. `Package's calculateCost` function should determine the cost by multiplying the weight by the cost per kilogram. Derived class `TwoDayPackage` should inherit the functionality of base class `Package`, but also include a data member that represents a flat fee that the shipping company charges for two-day-delivery service. `TwoDayPackage's` constructor should receive a value to initialise this data member. `TwoDayPackage` should redefine member function `calculateCost` so that it computes the shipping cost by adding the flat fee to the weight-based cost calculated by base class `Package's calculateCost` function.

Class `OverNightPackage` should inherit directly from class `Package` and contain an additional data member representing an additional fee per kilogram charged for overnight-delivery service. `OverNightPackage` should redefine member function `calculateCost` so that it adds the additional fee per kilogram to the standard cost per kilogram before calculating the shipping cost.

(i) Draw a class diagram depicting the three classes and their relationship. [3]
(ii) Write the C++ interface of each class [6]
(iii) Write the C++ implementation of each class [12]
(iv) Write a C++ program that creates objects of each type of package and test their member function `calculateCost`. [4]

END OF PAPER!!