UNIVERSITY OF SWAZILAND

FACULTY OF SCIENCE AND ENGINEERING

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

SUPPLEMENTARY EXAMINATION July 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 seven (7) pages including this page.

Question 1

a) Define encapsulation. [2]

b) How does overriding relate to polymorphism? [3]

c) Explain the difference between the use of the dot selection operator (.) and the arrow

member selection operator (->). [2]

d) What is a friend function of a class? [2]

e) What is a static class member? [2]

f) Why is it that static class members do not have the *this* pointer? [2]

g) Discuss four restrictions on operator overloading **in** C++. [4]

h) Explain why a class might provide a *set* and *get* functions for a data member? [2]

i) Explain 3 ways in which the members of a class can be accessed in the class's clients? [3]

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

Question 2

a) 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]

b) Explain the following Object-Oriented programming terms:
   a. Polymorphism [2]
   b. Class Constructor [2]
   c. Interface [2]

c) Using examples, discuss four ways by which class templates and inheritance are related. [4]

d) Discuss two problems of programming with the `switch` logic. Using an example, explain how polymorphism can be an effective alternative to `switch` logic. [4]

e) Explain the advantage of separating interface from implementation of a class. [2]

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

g) Using an example, explain where you would use a unary scope resolution operator. [2]

Question 3

(a) Using **only** *recursive* functions (NO repetition statements), write a C++ program that displays the following checkerboard pattern: [15]

```
* * * * * * * * * *
* * * * * * * * * *
  * * *     * * *
    * * * * * *
      * * * *
      * * * *
    * * * * * *
  * * *     * * *
* * * * * * * * * *
* * * * * * * * * *
```

Your program must use **only** three output statements, one (or more) of each of the following forms:

```
cout << "* ";
cout << " ";
cout << endln;
```

(b) Analyze the following program and determine its output. Show all you working.[10]

```
#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;
 }
```

Question 4

Create a class called `Invoice` that a hardware store might use to represent an invoice for an item sold at the store. An `Invoice` should include four pieces of information as data members - a part number (type string), a part name (type string), a quantity of the item being purchased (type double), a price per item (type double), and a value added tax (VAT) in percent (type double). Your class should have a constructor that initializes the four data members. Provide set and get functions for each data member. In addition, provide a member function named *InvoiceAmount* that calculates the invoice amount (i.e. multiplies the quantity by the price per item and then add the VAT), then returns the amount as a double value. If the quantity is not positive it should be set to 0.0. If the price per item is not positive it should be set to 0.0.

(i) Write an interface and implementation for this class. [20]

(ii) Write a test program that creates two `Invoice` objects. The first object represent the purchase of two shovels (part # SHOOI01), each costing E250. The second object represent the purchase of nine bags of cement (part # CMOIOI2), each costing E65. Assume that the current VAT set by the government is 14%. Your test program should print the values of each invoice. [5]

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 çompany 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!!!