THE OPEN UNIVERSITY OF SRI LANKA
B.Sc DEGREE PROGRAMME: LEVEL 04
FINAL EXAMINATION: SEMESTER 1-2014/2015
CPU2242: OBJECT ORIENTED PROGRAMMING USING C++ AND JAVA

DURATION: **THREE HOURS** (3 HOURS)

Date: **07th May, 2015** Time: **1.00 pm – 4.00 pm**

Answer FOUR Questions ONLY.

Q1.

a) Describe three (03) differences between procedure-oriented programming and object-oriented programming.

b) Define the following terms in object oriented programming.

i). Class
ii). Object
iii). Data abstraction
iv). Encapsulation

c) Differentiate between *private*, *protected* and *public* access specifiers.

d) State whether the following statements are TRUE or FALSE.

i). Java was invented by James Gosling who was the leader of a team from Sun Microsystem.

ii). Member functions defined inside a class definition are public by default in C++.

iii). In a class definition, data or functions designed protected are accessible only to member functions of that class.

iv). Classes are useful because they are removed from memory when not in use.

v). A constructor is executed automatically when an object is created.

Q2.

a) List three (03) differences between C++ and Java.

b) Explain the process of converting a Java stand-alone program (source code) into a machine language.

c) What is an *exception*? How is it handled in Java?

d) Consider the following Java class.

```
class Hello {
   public static void main (String[] args) {
     System.out.println ("Hello: This is my first Java
     program");
   }
}
```

Explain why the keywords **public**, **static** and **void** are included in the header of the main () method.

e) State whether the following names are valid Java identifiers. Briefly explain how you arrived to your conclusion.

i). for

ii). 2dayRainfall

iii). endSales

iv). weighted-average

v). my_salary

Q3.

a)

i). Why is *destructor* function required in a class in C++?

ii). Give two (02) characteristics of destructor.

b) Explain the difference between *constructor* and *copy constructor* in C++.

c) Explain briefly which of the following statement is calling for the copy constructor.

```
Date D1;
Date D1 (int n);
Date D1 (D2); //where D2 is an object of class Date
```

d)

i). Explain the order in which constructors are executed when there is multiple inheritance.

ii). Precisely write down the output of the following program.

```
#include <iostream>
using namespace std;

class a {
public:
    a() {
```

```
        cout<<"constructor a \n";
        }
        ~a() {
        cout<<"destructor a \n";
        }
};

class b : public a {
public:
        b() {
        cout<<"constructor b \n";
        }
        ~b() {
        cout<<"destructor b \n";
        }
};

int main() {
        b b1;
        return 0;
}
```

e) Write a C++ program to create a class named **Student** which includes the following data members and member functions.

Data members:
- Name of the student
- Index no.

Member functions:
- Default constructor
- Parameterized constructor
- Destructor
- Getter functions

Q4.

a) What is the difference between *function overloading* and *function overriding*?

b)

i). What is *operator overloading*?

ii). List three (03) C++ operators which cannot be overloaded.

c) State whether the following statements are **TRUE** or **FALSE**.

i). All C++ operators can be overloaded for user-defined classes.

ii). Operator overloading is available in Java.

iii). The ( ) and >= operators can be overloaded.

iv). A member function will have no arguments for unary operators and only one argument for binary operators.

v). To overload a postfix increment operator as a member function, it takes one argument.

d) Consider the following class definition in C++, which includes three data members. Rewrite the class by including the functions given from i) to iii).

```
class space{
   private:
        int x;
        int y;
        int z;
}
```

i). Parameterized constructor to initialize data members of the class.

ii). Overload the unary minus (-) operator using a member function or friend function.
(**Hint:** Unary minus operator; negates an expression. i.e. C = -A)

iii). Write a suitable main function to test your class.

Q5.

a) What is *inheritance*?

b) What is the difference between *single inheritance* and *multiple inheritance*? Explain using an example.

c) Fill the following table for visibilities of inherited members of a derived class in C++.

Base class visibility	**Derived class visibility**	
	Public derivation	**Private derivation**
Private		
Protected		
Public		

d) Consider the following class definitions.

```
class Parent1 {
 private:
   int i;
 protected:
   float x;
 public:
   double y;
   int getter_x ();
};
```

```
class Parent2 {
 private:
   int a;
 protected:
   float b;
 public:
   double z;
   int getter_a ();
};
```

```
class Child: private Parent1, public Parent2 {
 private:
   int c;
 public:
   int getter_c ();
   void print ();
};
```

Using the table filled in part Q5. c), find the visibilities of inherited members of the derived class **Child**.

e) Write the following classes in C++:

i). The class **Person** having data members: *name* and *age*.

ii). **Student** is a derived class from the base class **Person** having two data members: *index number* and *marks*

iii). **Employee** is a derived class from the base class **Person** having two data members: *empcode* and *designation*.

iv). Each of the classes should include the following member functions:

a. A default constructor and parameterized constructor

b. Getter functions for accessing the data members

Q6.

a) Explain the meaning of *polymorphism*.

b) Consider the following program in C++.

```
#include <iostream>
using namespace std;

class Shape { // Base class
public:
    void show (){
        cout << "I am a shape object\n";
    }
};

class Rectangle: public Shape {
public:
    void show (){
        cout << "I am a rectangle object\n";
    }
};
class Circle: public Shape {
public:
    void show (){
```

```
        cout << "I am a circle object\n";
    }
};

class Triangle: public Shape {
public:
    void show (){
        cout << "I am a Triangle object\n";
    }
};

int main() {
    Shape * ptr[3];
    ptr[0] = new Rectangle;
    ptr[1] = new Circle;
    ptr[2] = new Triangle;

    for (int i=0; i<3; i++){
        ptr[i]->show();
    }
    return 0;
}
```

Answer the questions given below.

i). What is the output of the above program?

ii). The user wants to display the output as follows. Rewrite the appropriate class to achieve the target of the user.

```
I am a rectangle object
I am a circle object
I am a Triangle object
```