THE OPEN UNIVERSITY OF SRI LANKA
B.SC. DEGREE PROGRAMME: LEVEL 04
DEPT. OF MATHEMATICS AND COMPUTER SCIENCE
FINAL EXAMINATION - 2011
CPU2242: OBJECT ORIENTED PROGRAMMING USING C++ AND JAVA

DURATION: THREE HOURS (3 HOURS)

Date: 28th December 2011 Time: 1.00pm – 4.00pm

Answer FOUR questions only.

1)

a) State whether the following statements are **TRUE** or **FALSE** and justify your answer.

i) In Java **Double** is a valid variable name while **double** is an invalid variable name.
ii) **int $_SESSION;** is an invalid variable declaration in Java.
iii) The following method definition is valid in java.

```
public String getName(){
        String name;
        return name +" is your name";
}
```

iv) In Java **protected members** of a class can only be accessed by inherited classes.
v) Arrays can hold data elements of different data types.

b) What is the most suitable data types (in Java) to represent each of the following items? By providing the suitable variable declarations briefly explain the reason to select the particular data type for each item.

i) Gender of a person.
ii) Average rainfall of a month.
iii) Number of executive employees in a company.
iv) State of a switch (on or off).
v) Planck's constant (6.626068×10^{-34}).

c) Briefly explain **recursion** and **base case** of a recursive method.

The **factorial** of a non-negative integer n, denoted by $n!$, is the product of all positive integers less than or equal to n (E.g. 5! = 5 x 4 x 3 x 2 x 1 = 120 and the value of 0! is 1).

i) Write a **recursive method** to find the factorial of a given non-negative integer n.

ii) Using the method you defined in part (i) write a new method to represent the following mathematical formula.

$$^{n}C_{r} = \frac{n!}{r!(n-r)!}$$

2)

a) Define the terms **Object, Class, ADT** (Abstract Data Type) and **Abstraction** in the context of Object Oriented Programming **(OOP).**

b) What do you mean by **Encapsulation?** Explain using an example.

c) Define a class to represent **Employee** with attributes **employeeName, employeeId, employeeAge, employeeDesignation** and **employeeBasicSalary.**

d) Define methods to set values for the attributes mentioned in **part (c).**

e) Write a method which will return employee's (approximate) age when year of birth is passed as a parameter.

f) When calculating the employee's net salary, in addition to the basic salary each employee gets Rs. 5,000.00 and 50% (half of basic salary) bonus. Bonus is added to the salary only if current month is April or December. Write a method to obtain employee's net salary when the basic salary and the current month are passed as parameters.

3) Class **Pet** consists of **Mammal** class and **Bird** class. **Mammal** class consists of **PetMammalCat** and **PetMammalDog** classes and **Bird** class consists of **PetBirdParrot** and **PetBirdCanary**. For all pets **petName** and **petAge** are common attributes. **Mammal** and **Bird** classes have additional attributes **mammalId** and **birdId** respectively. Further each pet has a unique attribute for favorite food (i.e. **favoriteCatFood, favoriteDogFood, favoriteParrotFood** and **favoriteCanaryFood**).

a) What do you mean by inheritance? What is the purpose of using inheritance in OOP? Discuss your answer by giving a classification (inheritance) hierarchy to represent the above information.

b) Write down suitable class definitions for the classification hierarchy defined in **part (a).** Each class should have methods to initialize and display information.

c) Write another class called **MainClass** with a proper main method to test the classes you defined in **part (b).**

4)

a) What do you mean by **Overloading** in the context of OOP? What types of overloading are you aware of? Discuss your answer using method prototypes.

b) Define a default constructor to a class called **MathFunction** and overload it with **numberOne , numberTwo** parameters.

c) Define appropriate methods for **MathFunction** class (i.e. addition, subtractions, multiplication and division). Use **numberOne** and **numberTwo** as parameters for your methods.

d) Overload your **addition** and **multiplication** methods with additional parameter called **numberThree .**

e) Using methods defined in **part c)** complete the **MathFunction** class so that one can enter vales of attributes (numberOne, numberTwo) and calculate addition, subtraction, multiplication and division for a particular instance of **MathFunction** class.

5)

a) What is an abstract class? Give an example.

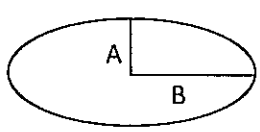
b) **"In OOP polymorphism can be achieved via method overloading and method overriding"** Discuss using an example.

c) The **Shape** class cannot be instantiated and consists of shapes **Triangle, Rectangle** and **Ellipse. Square, right angle triangle** and **circle** are derived shapes of Rectangle, Triangle and Ellipse respectively.

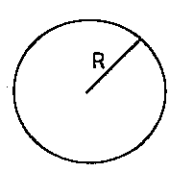
i) Draw a suitable classification (inheritance) hierarchy to represent the above information. Clearly mention if there are any abstract classes.

ii) Write suitable class definitions for the classification hierarchy you defined in part(i). Each class should facilitate two methods to calculate the area (decide suitable parameters according to the equations given below) of a particular shape and display the area (use overloading and overriding).

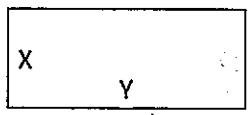
Area of an ellipse = π*A*B



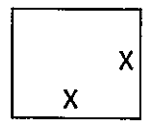
Area of a circle = π*R*R



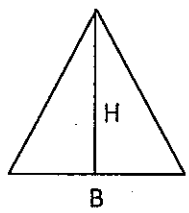
Area of a rectangle = X*Y



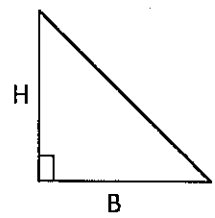
Area of a square = X*X



Area of triangle = (B*H)/2



Area of a right angle triangle = (B*H)/2



6)

a) What is a Java interface? What are the major advantages of Java interfaces?

b) Briefly discuss the similarities and differences between an **abstract class** and an **interface.**

c) The interface **calculator** consists of **standard** calculator and **scientific** calculator. The methods *addition, subtraction, multiplication* and *division* are common to both calculators. That is both standard and scientific calculators can do basic math functions. In addition to the basic math functions scientific calculators have methods to calculate *square root, sin, cos* and *tan*. Both standard calculator and scientific calculator cannot be instantiated and methods are not defined inside scientific calculator and standard calculator.

Write a complete Java program to satisfy the above description and your program should facilitate the followings.

i) The basic math methods should be able to call without creating class objects. That is basic math methods are class methods.

ii) The *standard calculator* and *scientific calculator* classes cannot be extended.