SET-A

SRM INSTITUTE OF SCIENCE &TECHNOLOGY,
RAMAPURAM CAMPUS,
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS LEARNING ASSESSMENT-2
18CSC202J– Object Oriented Design and Programming

Branch : B.Tech CSE & All Specializations
Year / Sem : II / III
Duration : 90 Minutes
Max. Marks: 50
Date of Exam:

Part-A (10*1=10 Marks)

1. **Which inheritance type is used in the class given below?**

   class A : public X, public Y

   a. Multilevel inheritanc
   b. Multiple inheritance
   c. Hybrid inheritance
   d. Hierarchical Inheritance

2. **Which of the following operators cannot be overloaded?**
   a. []
   b. ->
   c. ?:
   d. *

3. **Dynamic aspects related to a system are shown with help of**
   a. sequence diagrams
   b. interaction diagrams
   c. deployment diagrams
   d. use case diagrams

4. **Which class is used to design the base class?**
   a) abstract class
   b) derived class
   c) base class
   d) derived & base class

5. **_____diagram is time-oriented?**
   a. Collaboration
   b.Sequence
   c. Activity
   d.Statechart

6. **Which is also called as abstract class?**
   a) virtual function
   b) pure virtual function
   c) derived class
   d) base class

7. **Which is the correct syntax of defining a pure virtual function?**
   a) pure virtual return_typefunc();
   b) virtual return_typefunc() pure;
   c) virtual return_typefunc() = 0;
   d) virtual return_typefunc();

8. **Which is the correct statement about pure virtual functions?**
   a) They should be defined inside a base class
   b) Pure keyword should be used to declare a pure virtual function
   c) Pure virtual function is implemented in derived classes
   d) Pure virtual function cannot implemented in derived classes

9. **What is the syntax of friend function?**
   a)**friend class1 Class2;**
   b) friend class;
   c) friend class
   d) friend class()

10. **If same message is passed to objects of several different classes and all of those can respond in a different way, what is this feature called?**
    a) Inheritance
    b) Overloading
    c) Polymorphism
    d) Overriding

Part – B (4 X 4 = 16 Marks)

Answer any 4 Questions

11. When do we need constructor overloading?

To initialize data member of class: In the constructor member function (which will be declared by the programmer) we can initialize the default vales to the data members and they can be used further for processing.

To allocate memory for data member: Constructor can also be used to declare run time memory (dynamic memory for the data members).

12. Specify the restrictions on Operator overloading

Following C++ Operator can't be overloaded

- Class member access operators(.& .*)
- Scope Resolution Operator(::)
- SizeofOperator(sizeof())
- Conditional Operator(? :)
- Overloading restrictions
- Precedence ,associativity of an operator cannot be changed
- No new operators can be created
- Use only existing operators
- No overloading operators for built-in types
- Cannot change how two integers are added

13. List out the modes of inheritance with an example

3 Types of Access Specifiers

Public mode: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.

Protected mode: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.

Private mode: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class.

14. Write short notes on Virtual Function with an example

A C++ virtual function is a member function in the base class that you redefine in a derived class.

It is declared using the virtual keyword.

It is used to tell the compiler to perform dynamic linkage or late binding on the function.

Syntax for declaring a virtual function

```
classclass_name
 {
public:
 virtualreturn_typefunction_name()
};
```

Example:

```
class A
{ public:
virtual void fun_1()
};
```

15. What is an Activity diagram? List its benefits

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

- Activity diagram is used to represent the flow from one activity to another activity within the system rather than the implementation.
- It models the concurrent and sequential activities.
- This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

16. Define friend function with a suitable example

Friend functions and friend classes

- A friend function is a function that is declared outside a class, but is capable of accessing the private and protected members of class.
- Generally, The private members cannot be accessed from outside the class. i.e a non

member function cannot have an access to the private data of a class.

- In C++ a non member function can access private by making the function friendly to a class.
- Friend function is declared by using keyword "friend"

Characteristics of a Friend function:

- It can be invoked like a normal function without using the object.
- It cannot access the member names directly and has to use an object name and dot membership operator with the member name.
- It can be declared either in the private or the public part.
- Objects of the class or their pointers can be passed as arguments to the friend function.

Ex: friend float mean(Sample s), where sample is the class name in which then mean function is declared as friend

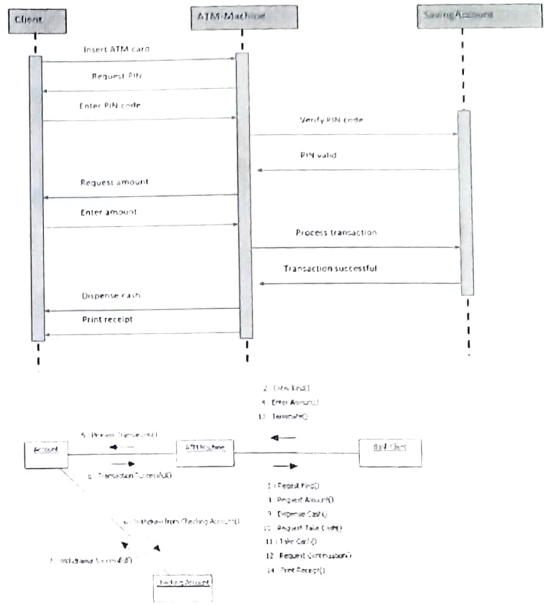
- The keyword friend is placed only in the function declaration, not in the function definition.

Syntax:

```
classclass_name
{
frienddata_typefunction_name(arguments/s);
};
```

Part – C (2 X 12 = 24 marks)

17 a. Discuss interaction diagram and Illustrate interaction diagram for withdrawing amount from ATM machines



(OR)

b. i) Explain the types of constructors with suitable examples (6 marks)

The constructor is automatically called when an object is created

There are several forms in which a constructor can take its shape namely:

*Default Constructor

*Parameterized Constructors

*Copy constructor

Default constructor is the constructor which doesn't take any argument. It has no parameter,also Called as Empty Constructor

- It may be necessary to initialize the various data elements of different objects with different values when they are created.
- This is achieved by passing arguments to the constructor function when the objects are created.
- The constructors that can take arguments are called parameterized constructors.

ØA copy constructor is used to declare and initialize an object from another object.
ØA reference variable has been used as an argument to the copy constructor.
ØWe pass the object of a class into another object of same class,
ØThis is used for copying the values of class object into another object of class so that we can call them as copy constructor.

ii) Write a C++ program for constructor overloading (6 marks)

```
#include <iostream>
using namespace std;
class Person
{
 private:
int age; // data member
public:
   // 1. Constructor with no arguments
Person()
   {
age = 20; // when object is created the age will be 20
   }
   // 2. Constructor with an argument
Person(int a)
{ // when parameterised Constructor is called with a value the
      // age passed will be initialised
age = a;
   }
intgetAge()
{ // getter to return the age
return age;
   }
};
int main()
{
   Person person1, person2(45); // called the
object of person class in differnt way

cout<<"Person1 Age="<< person1.getAge() <<
endl;
cout<<"Person2 Age = " << person2.getAge()<<endl;
return 0;
}
```

Output:
Person1 Age = 20
Person2 Age = 45

18 a. Describe inheritance and its various types with example C++ Programs

- Single Inheritance
- Multi Level Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance
- Multiple Inheritance

```
#include <iostream>
#include <string>
using namespace std;
class Animal
{
   string name="";
   public:
   int tail=1;
   int legs=4;
};
class Dog : public Animal
{
   public:
   void voiceAction()
```

```
    {
      cout<<"Barks!!!";
    }
};
int main()
{
  Dog dog;
  cout<<"Dog has "<<dog.legs<<" legs"<<endl;
  cout<<"Dog has "<<dog.tail<<" tail"<<endl;
  cout<<"Dog ";
  dog.voiceAction();}
```

Output:

```
Dog has 4 legs
        Dog has 1 tail
        Dog Barks!!!
```

- We have a class Animal as a base class from which we have derived a subclass dog. Class dog inherits all the members of Animal class and can be extended to include its own properties, as seen from the output

(OR)

b. Elaborate statechart diagram online shopping system

R. Seth
Course Coordinator
CR. SATHYA

HOD/CSE