submitted by: Mahesh Singh Madai

#0

Object Oriented Programming

Solution Sets

#1. 2075

Discuss the feature of Object-Oriented Programming. Differentiate between Object Oriented programming and any other programming languages you know.

The features of OOP are as follows:

a) Objects: Objects are the entities in an object oriented system through which we can perceive the world around us. The entities are then represented as objects in the program. They may represent a person, place, bank account or any items that the program must handle

b) Classes:

Class is the collection of objects of similar type. It defines a data type, much like a struct in C. and built-in data type. It defines what data and functions will be included in objects of that class. Each class encapsulates the data and functions.

c) Encapsulation and Data Abstraction:

The wrapping up of data and function into a single unit is called encapsulation is most striking of a class. The data isnot accessible from outside of the class. Only member function can access data on that class. The

Scanned by CamScanner

insulation of data from direct access by the program is called data hiding.

d) Inheritance:

Inheritance is the process by which objects of one class acquire the characters of objects of another class. In OOP, the concept of inheritance provides the idea of reusability. You can use additional features to an existing class without modifying it.

e) Polymorphism:

The polymorphism allows different objects to respond to the same message in different ways the response specific the type of objects. It is important when OOP is dynamic creating and destroying the objects in run-time.

Differences between Object Oriented Programming and Procedure Oriented Programming are as follows:

Procedure Oriented Programming	Object Oriented Programming
i) Program is divided into small parts called functions.	i) Program is divided into p called classes
ii) It follows top down approach.	ii) It follows bottom up app
Data is not encapsulated	iii) Data is encapsulated.

Scanned by CamScanner

iv) If does not have only access specifier.

iv) OOP has access specifier named public, private, protected

v) Overlooding isnot possible

eg: C, VB, Pascal etc

v) Overloading is possible.

eg: C++, JAVA, C#, etc

3. Create a class stack with suitable data members and member functions to push and pop the elements of the stack. Add the exception, when user tries to add item while the stack is full and when user tries to delete item while stack is empty. Throw exception in both of the cases and handle these exceptions

Scanned by CamScanner

4. 'Concept of friend function is against the philosophy of Object Oriented Programming'. Explain.

⇒ The concept of encapsulation and data hiding d that non-member functions should not be allowed to access an object private and protected members. Sometimes, this feat leads to considerable inconvenience in programming.
By using friend function, we can access the data of the particular class in which it is defined

eg:
```
class Test{
  private:
      int data;
  public:
      friend void show();
};
void show()
{
   Test t;
   t.data=25;
   cout<<t.data<<endl;
}
void modify
{
   show();
}
```

Scanned by CamScanner

Explain the importance of constructor and destructor.

Constructor is a 'special' member function whose task is to initialize the objects of the class. It has same name as the class. The constructor is invoked whenever an object of its associated class is created. It constructs the values of data members of the class.

When a class contains a constructor, it is guaranteed that an object created by the class will be initialised automatically.

```
class Integer
{
    int m,n;
  public:
    integer(void);
    ——
};
Integer::Integer(void)
{
    m=0; n=0;
}
```

The declaration integer int1 ,not only creates object int1, but also initialises its data members m and n. There is no need to write statements to invoke constructor function.

Destructor is used to destroy the memory space of objects that have been created by a constructor.

First a constructor is executed and then a destructor.

Scanned by CamScanner

What is template? How can you differentiate a function template from a class template?

Template is a feature that enables us to define generic classes and functions and thus provid support for generic programming. A template is used to make generic classes that can operate upon various data types and no separate data types has to be listed

When an object of a specific type is defined the template definition for that class is substituted with the required data type. Template enables the programmer to create a class or function of any data type instead of defining new class/ function every time.

Class Template and a Function Template:-

class template is used to create data and functions whereas function template is used to create family of templates.

The function template syntax is similar to that of class templ except that we are defining functions instead of classes

```
template<class T>
return_type functionname(arguments)
{
  ==
  ==  //Body
}
```

```
template<class T>
class classname
{
  ---
  ---
  ---
};
```

Scanned by CamScanner

7) Explain about 'this' pointer with suitable example.

⇒

In C++, there is a keyword 'this' to represent an object that invokes a member function; 'this' pointer is a pointer that points to the object for which 'this' function was called.

For example: a function call A.max() will set the pointer this to the address of the object A. The starting address is the same as the address of the first variable in the class structure.

This unique pointer is automatically passed to a member function when it is called. 'this' pointer acts as an implicit argument to all the member function.

Example:

```
class Box
{
  public:
    Box(double l=2.0, double b=2.0, double h=2.0) {
       length = l;
       breadth = b;
       height = h;
    }
    double Volume() {
       return length * breadth * height;
    }
    int compare (Box box) {
       return this -> volume() > box.volume();
    }
```

Scanned by CamScanner

```
private:
    double length;
    double breadth;
    double height;
};
```

8) Write a c++ program containing a possible exception. Use a block to throw it and a catch block to handle it.

Ans

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

double division(int a, int b)
{
    if(b==0)
      {
        throw "Division by zero condition!";
      }
    return(a/b);
}
int main()
{
   int x=50;
   int y=0;
   double z=0;
   try{
       z=division(x,y);
       cout<<z<<endl;
   }
```

Scanned by CamScanner

```
catch (const char * msg)
{
    cout<<msg<<endl;
}
return 0;
}
```

9) Differentiate between compile time and run time polymorphism.

	Compile time Polymorphism	Run time Polymorphism
i)	In compile time call is resolved by the compiler	i) In this call is not resolved by the compiler
ii)	Known as static / early binding	ii) It is known as dynamic / late binding.
~~iii)~~	~~An obj~~	
iii)	An object is bound in its function at compile time	iii) Selection of the appropriate function is done at run time
iv)	It doesnot require the use of pointers.	iv) It requires the use of pointer.
v)	~~less~~ Greater efficiency	v) Less efficiency
vi)	Function call are faster.	vii) Function call execution are slower
	eg: function overloading	eg: virtual function.

Scanned by CamScanner

10) What is a container class? Differentiate container class from inheritance.

Ans A container class is a class designed to hold and organize multiple instances of another type. These container classes typically implement a fairly standardized minimal set of functionality. They implement a member-of relationship

eg: ~~lists, vector and strings~~.

A container class is a class that holds group of same or mixed objects in memory. It can be hetergeneous and homogeneous.

Meanwhile, Inheritance is the ability for a class to inherit properties and behaviour from a parent class by extending it.

whereas,

Containership is the ability of a class to contain objects of different class as member data. In simple terms, a class has been called in another class.

```
class A
{ --} obj1;
class B
{
  A obj1;
}
```

containership

```
class A
{
  --};
class B : public A
{ --};
```

Inheritance.

Scanned by CamScanner

13) Define the various ambiguity situations that may occur during the process of inheritance. How can you resolve the ambiguity situation?

Ans Ambiguity cases arises in inheritance when the same name appears in more than one base class.

```
class M
{
  public:
     void display(void)
     {
        cout<< "class M\n";
     }
};
class N
{
   public:
     void display(void)
     {
        cout<< "class N\n";
     }
};
```

When display() function is used, the derived class is invoked and operation is done

To solve this, we can use scope resolution operator (class) as:

```
{
   M::display();
}
```

Scanned by CamScanner

Another way is to solve by using virtual base class, as below

```
class A
{ public:
    int a;
};
class B : virtual public A
{ public:
    int b;
}
void main()
{
    B obj;
    obj.a = 10
    obj.a = 100;

}
```

Scanned by CamScanner