Introduction to Object Oriented Programming

Unit – 1

Short Questions

1. Define Encapsulation.
2. Define Data Abstraction.
3. Define Inheritance.
4. Define Multiple Inheritance.
5. Define Polymorphism.
6. Define Message Passing.
7. Define Extensibility.
8. Define Persistence.
9. Define Delegation.
10. Define Containership.
11. Define Genericity.
12. Define Abstract Data types.
13. Define Objects.
14. Define Classes.
15. Define dynamic binding.
16. What is procedure oriented programming?
17. What is object oriented programming?

Long Questions

1. Discuss the syntactical differences between C and C++.
2. Why does the object-oriented philosophy need functions to be defined inside the classes? What could be the advantage?
3. What is software crisis? Justify the need for a new programming paradigm. Explain how object paradigm overcomes this software crisis.
4. Pick up any C program that you have written earlier. Rename that program as ".cpp" program. Try to compile and run it. List out the problems that you have faced during the execution of that program.
5. What is object oriented paradigm? Explain the various features of OO paradigm.
6. What are the programming paradigms currently available? Explain their features with programming languages supporting with them.
7. Compare structured and OO programming paradigm.
8. What are the elements of object oriented programming? Explain its key components like objects and classes with example.
9. Write an object representation (pictorial) of student class.
10. Explain multiple views of an object with suitable example.
11. What is a difference between inheritance and delegation? Illustrate with example.
12. List different methods of realizing polymorphism and explain them with example.
13. What are the steps involved in OO programming? Explain its message communication model.
14. Which is the first object oriented language? Explain the heritage of C++.

15. Explain the merits and demerits of Object Oriented methodology.
16. What is software reuse? What is difference between reuse and porting? What are the factors influencing the software reuse?
17. Identify the reusable components in software and discuss how OOPs helps in managing them.
18. Justify "Class is a template while Object is data".
19. Discuss how object-oriented paradigm affects different elements of computing such as hardware architecture.
20. What is procedure-oriented language? What are the main characteristics?
21. Distinguish between the following terms.
    a. Objects and Classes
    b. Data abstraction and Data encapsulation.
    c. Inheritance and polymorphism.
    d. Dynamic binding and message passing.
22. Describe inheritance as applied to OOP.
23. How does the object oriented approach differ from object based approach?

Multiple Choice Question

1. You can use C++ as a procedural, as well as an object-oriented, language
    a. True
    b. False
    c. May be
    d. None of above
2. Which of the following type of class allows only one object of it to be created?
    a. Virtual class
    b. Abstract class
    c. Singleton class
    d. Friend class
3. Which of the following is not a type of constructor?
    a. Copy constructor
    b. Friend constructor
    c. Default constructor
    d. Parameterized constructor
4. Which of the following statements is correct?
    a. Base class pointer cannot point to derived class.
    b. Derived class pointer cannot point to base class.
    c. Pointer to derived class cannot be created
    d. Pointer to base class cannot be created.
5. Which of the following is not the member of class?
    a. Static function
    b. Friend function
    c. Const function
    d. Virtual function
6. Which of the following concepts means determining at runtime what method to

invoke?
   a. Data hiding
   b. Dynamic Typing
   c. Dynamic binding
   d. Dynamic loading
7. Which of the following term is used for a function defined inside a class?
   a. Member Variable
   b. Member function
   c. Class function
   d. Classic function
8. Which of the following concept of oops allows compiler to insert arguments in a function call if it is not specified?
   a. Call by value
   b. Call by reference
   c. Default arguments
   d. Call by pointer
9. How many instances of an abstract class can be created?
   a. 1
   b. 5
   c. 13
   d. 0
10. Which of the following cannot be friend?
   a. Function
   b. Class
   c. Object
   d. Operation function
11. Which of the following concepts of OOPS means exposing only necessary information to client?
   a. Encapsulation
   b. Abstraction
   c. Data Hiding
   d. Data binding
12. Why reference is not same as a pointer?
   a. A reference can never be null.
   b. A reference once established cannot be changed.
   c. Reference doesn't need an explicit dereferencing mechanism.
   d. All of the above.
13. cout is a/an _____ .
   a. operator
   b. function
   c. object
   d. macro
14. Which of the following concepts provides facility of using object of one class inside another class?
   a. Encapsulation

b. Abstraction
c. Composition
d. Inheritance

15. How many types of polymorphisms are supported by C++?
    a. 1
    b. 2
    c. 4
    d. 3
16. Which of the following is an abstract data type?
    a. int
    b. double
    c. string
    d. class
17. Which of the following concepts means adding new components to a program as it runs?
    a. Data hiding
    b. Dynamic typing
    c. Dynamic binding
    d. Dynamic loading
18. Which of the following statement is correct?
    a. A constructor is called at the time of declaration of an object.
    b. A constructor is called at the time of use of an object.
    c. A constructor is called at the time of declaration of a class.
    d. A constructor is called at the time of use of a class.
19. Which of the following correctly describes overloading of functions?
    a. Virtual polymorphism
    b. Transient polymorphism
    c. Ad-hoc polymorphism
    d. Pseudo polymorphism
20. Which of the following approach is adapted by C++?
    a. Top-down
    b. Bottom-up
    c. Right-left
    d. Left-right

True False

1. In procedure-oriented programming, all data are shared by all functions.
2. The main emphasis of procedure-oriented programming is an algorithms rather than on data.
3. A constructor is called at the time of declaration of an object.
4. One of the striking features of object oriented programming is the division of programs into objects that represent real world entities.
5. Wrapping up of data of different types into a single unit is known as encapsulation.
6. One problem with OOP is that once a class is created it can never be changed.
7. Inheritance means ability to reuse the data values of one object by.

8. Polymorphism is extensively used in implementing inheritance.
9. Object-oriented programs are executed much faster than conventional program.
10. Object-oriented systems can scale up better from small to large.
11. Object-oriented approach cannot be used to create database.
12. Polymorphism is the process by which objects of one class acquire properties of objects of another class.
13. Base class pointer cannot point to derived class.
14. Pointer to derived class cannot be created.
15. A constructor is called at the time of declaration of a class.
16. Derived class pointer cannot point to base class.
17. A constructor is called at the time of declaration of an object.
18. Class data members are private by default while that of structure are public by default.
19. Class can have member functions while structure cannot.
20. Pointer to structure or classes cannot be declared.

Fill in the blanks

1. The "_____" keyword is used to create an instance of a class.
2. _____ concepts means wrapping up of data and functions together.
3. _____ concepts means waiting until runtime to determine which function to call.
4. Constructor cannot be used with the keyword _____.
5. _____ way of declaring a function as constant.
6. _____ concepts is used to implement late binding.
7. Binding refers to the linking of a _____ to the code to be executed in response to the call.
8. _____ are the basic run time entities in an OOP.
9. The wrapping up a data into a single unit called as a _____.
10. Inheritance supports concept of _____ classification.
11. The concept of inheritance provides the idea of _____.
12. The entire data and code of an object can be made a user-defined data type with the help of a _____.
13. Insulation of data from direct access by the program is called _____.
14. OOP employs the _____ approach.
15. The most important benefit of OOP over the conventional programming method is _____.
16. _____ means one name, multiple forms.

Object Initialization and Cleanup

Unit – 2

Short Questions

1. What is the need of explicit constructor?
2. When will the destructor be called?
3. What is the life-time of an object?
4. Define constructor.

5. Define copy constructor.
6. Define default constructor.
7. Define destructor.
8. What is dynamic initialization?
9. Define explicit constructor.
10. What is initialization of the object?
11. What is member initialization list?
12. What is parameterized constructor?
13. When destructor will get invoked?
14. What is constructor overloading?
15. Define nameless object.
16. Use of copy constructor.
17. What is constant object?
18. What is constant constructor?
19. What is nested class?

Long Questions:

1. What is dynamic initialization? Where it can be useful?
2. What is the use of scope resolution operator? Suggest an example of its use in practical circumstances. Does it have any use other than separating out global variables from local variables?
3. What are the advantages of new and delete operators over malloc() and free() functions.
4. What is the need for initialization of objects using a constructor? What could be the problems if constructors are not provided in C++?
5. What are the functions of constructor? How are they different from normal functions?
6. What is the difference between default constructor provided compiler and a user-define default constructor?
7. List the cases where default constructor provided by user becomes necessary.
8. What is meant by dynamic reinitializing an object? What is the difference between normal initialization and dynamic initialization?
9. We may need constructor in every class that we define; but we may need destructor in few classes only. Why?
10. What is the life-time of an object? What is the relation of the lifetime of an object to the constructor and destructor?
11. When we need the copy constructor in the definition of a class? Justify your answer.
12. What are the constructors and destructors? Explain how they differ from normal function.
13. What are the differences between default and parameterized constructors?
14. What are the copy constructors and explain their need?
15. What is the order of construction and destruction of object?
16. What are read only objects? What is the role of constructor in creating such objects?
17. What is need for initialization of objects using a constructor? What could be the

problem if constructor is not provided in C++?

18. What are the functions of constructor? How are they differing from normal function?
19. What is the difference between a default constructor provided by the compiler and a user-define default constructor?
20. List the cases where default constructor provided by user becomes necessary?
21. What is the need of explicit constructor?
22. What is the difference between parameterized constructor and default constructor?
23. What is the advantage of default argument in a constructor?
24. Give an example where if we do not provide copy constructor, behavior of the program becomes unacceptable.
25. What is constructor? Is it mandatory to use constructor in a class?
26. List some of the special properties of the constructors.
27. Can we have more than one constructor in a class? If yes then explain the need for such a situation.
28. What do you mean by dynamic initialization of objects? Why do we need to do this?
29. How is dynamic initialization of object achived?
30. Distinguish between the following two statements:
    time T2(T1);
    time T2=T1;
    T1 and T2 are objects of time class.
31. Describe the importance of destructors.

Multiple Choice Questions

1. A constructor that accepts _____ parameters is called the default constructor.
   a. One
   b. Two
   c. Three
   d. No
2. What happens when a class with parameterized constructors and having no default constructor is used in a program and we create an object that needs a zero-argument constructor?
   a. Compile-time error.
   b. Preprocessing error.
   c. Runtime error.
   d. Runtime exception.
3. Can a class have virtual destructor?
   a. Yes
   b. No
   c. May be
   d. None of above
4. Destructor has the same name as the constructor and it is preceded by
   a. !
   b. ?

c. ~
d. $

5. For automatic objects, constructors and destructors are called each time the objects
   a. enter and leave scope
   b. inherit parent class
   c. are constructed
   d. are destroyed
6. Which constructor function is designed to copy objects of the same class type?
   a. Create constructor
   b. Object constructor
   c. Dynamic constructor
   d. Copy constructor
7. Which of the following statement is correct?
   a. Constructor has the same name as that of the class.
   b. Destructor has the same name as that of the class with a tilde symbol at the beginning.
   c. Both A and B.
   d. Destructor has the same name as the first member function of the class.
8. Which of the following statement is incorrect?
   a. Constructor is a member function of the class.
   b. The compiler always provides a zero argument constructor.
   c. It is necessary that a constructor in a class should always be public.
   d. Both B and C.
9. When are the Global objects destroyed?
   a. When the control comes out of the block in which they are being used.
   b. When the program terminates.
   c. When the control comes out of the function in which they are being used.
   d. As soon as local objects die.
10. Copy constructor must receive its arguments by _____ .
    a. either pass-by-value or pass-by-reference
    b. only pass-by-value
    c. only pass-by-reference
    d. only pass by address
11. A function with the same name as the class, but proceeded with a tilde character (~) is called _____ of that class.
    a. Constructor
    b. Destructor
    c. Function
    d. Object
12. A union that has no constructor can be initialized with another union of _____ type.
    a. different
    b. same
    c. virtual
    d. class

13. Which of the following gets called when an object goes out of scope?
    a. Constructor
    b. Destructor
    c. Main
    d. virtual function
14. Which of the following statement is correct?
    a. Destructor destroys only integer data members of the object.
    b. Destructor destroys only float data members of the object.
    c. Destructor destroys only pointer data members of the object.
    d. Destructor destroys the complete object.
15. _____ used to make a copy of one class object from another class object of the same class type.
    a. Constructor
    b. Copy constructor
    c. Destructor
    d. Default constructor
16. Constructor _____ to allow different approaches of object construction.
    a. cannot overloaded
    b. can be overloaded
    c. can be called
    d. can be nested
17. Which of the following statement is correct?
    a. A destructor has the same name as the class in which it is present.
    b. A destructor has a different name than the class in which it is present.
    c. A destructor always returns an integer.
    d. A destructor can be overloaded.
18. Which of the following cannot be declared as virtual?
    a. Constructor
    b. Destructor
    c. Data Members
    d. Both A and C
19. If the copy constructor receives its arguments by value, the copy constructor would
    a. call one-argument constructor of the class
    b. work without any problem
    c. call itself recursively
    d. call zero-argument constructor
20. Which of the following are NOT provided by the compiler by default?
    a. Zero-argument Constructor
    b. Destructor
    c. Copy Constructor
    d. Copy Destructor

True False

1. Constructors, like other member functions, can be declared anywhere in the class.
2. Constructor does not return any value.

3. A constructor that accepts no parameter is known as the default constructor.
4. A class should have at least one constructor.
5. Destructor never takes any arguments.
6. Constructor must be explicitly invoked.
7. Constructor defined in private section is useful.
8. Constructor can return value.
9. Destructors are invoked automatically.
10. Destructor takes input parameters.
11. Destructor can be overloaded.
12. Constructor cannot be overloaded.
13. Constructor can take default arguments.
14. Data members of nameless objects can be initialized using constructors only.
15. Constructors can allocate memory during runtime.
16. A class member function can take its class's objects as value arguments.
17. Constant object can be initialized by using constructors only.
18. Data members of a class can be initialized by using constructors only.

Fill in the blanks

1. A constructor has the same name as that of _____.
2. Constructors are normally used to _____.
3. When an object is created and initialized at the same time, a _____ constructor gets called.
4. _____ used to destroy the objects that have been created by a constructor.
5. _____ never takes any arguments nor does it return any value.
6. To free allocated memory _____ keyword used.
7. To allocate memory to any variable _____ keyword used.
8. Allocation of memory to objects at the time of their construction is known as _____.
9. A _____ takes a reference to an object of the same class as itself as an argument.
10. A _____ is a special member function whose main operation is to allocate the required resources such as memory and initialize the objects if its class.
11. Constructor with arguments is called _____.
12. _____ is invoked when an object is destroyed.
13. Constructors is that a class can have multiple constructor is called _____.
14. Destructor can be _____.
15. _____ cannot be virtual.
16. If no constructors are defined, the compiler tries to generate a _____ constructor.
17. In _____ the parameter of a constructor can be of any of the data types except an object of its own class as a parameter.

Operator Overloading and Type Conversion

Unit – 3

Short Questions

1. What is the significance of operator overloading?
2. What are function object?
3. List down the operators that cannot be overloaded as a friend.
4. Define operator overloading.
5. Define operator.
6. Define unary operator.
7. Write any one step for the process of overloading.
8. Define friend function.
9. State different unary operator.
10. Define binary operator.
11. Example of binary operator.
12. State any 2 rules for overloading operator.
13. What is type conversion?
14. State three different type conversion.
15. Define basic to class type conversion.
16. Define class to basic type conversion.
17. Define one class to another class type conversion.
18. State any one condition for the casting operator function that should be satisfied by user.
19. List the operator that cannot be overloaded.
20. What is operator function?
21. Example of basic to class type conversion.

Long Questions

1. Differentiate between overloading of unary operators and overloading of binary operators.
2. Differentiate between overloading postfix operator and prefix operator.
3. Overloading new and delete may help us in managing memory ourselves. Explain.
4. What are the different types of conversion? Compare them.
5. What is operator overloading? Explain the importance of operator overloading.
6. Why it is necessary to overload an operator?
7. What is an operator function? Explain it with syntax and example.
8. How many arguments are required in the definition of an overloaded unary operator?
9. A class alpha has a constructor as follow:
   alpha(int a, double b);
   Can we use this constructor to convert types?
10. What is a conversion function? How is it created? Explain its syntax.
11. A friend function cannot be used to overload the assignment operator =. Explain why?
12. When is a friend function compulsory? Give an example.
13. We have two classes, X and Y. if a is an object of X and b is an object of Y and we want to say a=b; what type of conversion routine should be used and where?
14. Identify the error from the following code and also explain how we can overcome that error.

```
#include <iostream.h>
Class Space
{
intmCount;
public:
    Space()
    {
mCount=0;
    }
    Space operator ++()
    {
mCount++;
        return Space(mCount);
     }
};
Void main()
{
    Space objSpace;
objSpace++;
}
```

15. What are the limitations of overloading unary increment/decrement operator? How are they overcome?
16. Explain the syntax of binary operator overloading.
17. How many arguments are required in the definition an overloaded binary operator?
18. Write a program to overload unary operator for processing counters. It should support both upward and downward counting.
19. Design classes called polar and rectangle for representing a point in a polar and rectangle systems. Support data conversion function to support statements such as:
    Rectangle r1, r2; Polar p1,p2;
    r1=p1; p2=r2;
20. Why friend function not allowed to access members of a class directly although its body can appear within the class body?
21. Suggest and implement an approach to trace memory leakage.
22. State rules for overloading operators.
23. What is type conversion? Explain different types of type conversion supported in C++.

Multiple Choice Question

1. Which of the following is not an operator overloaded by the C++ language?
   a. pow
   b. >>
   c. +
   d. <<

2. To use an operator on user-defined class objects, operator overloading:
   a. must never be used, with three exceptions.
   b. must never be used.
   c. must always be used.
   d. must always be used, with three exception.
3. The correct function name for overloading the addition (+) operator is:
   a. operator_+
   b. operator:+
   c. operator+
   d. operator(+)
4. Which of the following operators cannot be overloaded?
   a. The . operator
   b. The -> operator
   c. The [ ] operator
   d. The & operator
5. Which statement about operator overloading is false?
   a. New operators can never be created.
   b. Certain overloaded operators can change the number of arguments they take.
   c. The precedence of an operator cannot be changed by overloading.
   d. Overloading cannot change how an operator works on built-in types.
6. To implicitly overload the += operator:
   a. Only the = operator needs to be overloaded.
   b. Only the + operator needs to be overloaded.
   c. The += operator cannot be overloaded implicitly.
   d. Both the + and = operators need to be overloaded.
7. Which of the following operators can be overloaded as a global function?
   a. ()
   b. ==
   c. +=
   d. []
8. Which situation would require the operator to be overloaded as a global function?
   a. The left most operand must be a class object (or a reference to a class object).
   b. The left operand is an int.
   c. The operator returns a reference.
   d. The overloaded operator is =.
9. An overloaded + operator takes a class object and a double as operands. For it to be commutative (i.e., a + b and b + a both work):
   a. The + operator cannot be overloaded to be commutative.
   b. operator+ must be a non-member function.
   c. operator+ must be a member function of the class from which the objects are instantiated.
   d. It must be overloaded twice; the operator+ function that takes the object as the left operand must be a member function, and the other operator+ function must be a global function.
10. Suppose you have a programmer-defined data type Data and want to overload the

<< operator to output your data type to the screen in the form cout<<dataToPrint; and allow cascaded function calls. The first line of the function definition would be:
   a. ostream&operator<<(const Data &dataToPrint, ostream&output).
   b. ostream&operator<<(ostream&output, const Data &dataToPrint)
   c. ostream operator<<(ostream&output, const Data &dataToPrint).
   d. ostream operator<<(const Data &dataToPrint, ostream&output).
11. Suppose the unary !operator is an overloaded member function of class String. For a String object s, which function call is generated by the compiler when it finds the expression !s?
   a. A compiler error results because no arguments are given.
   b. operator!(s)
   c. s.operator!(default_value1, default_value2,…).
   d. s.operator!().
12. y and z are user-defined objects and the += operator is an overloaded member function. The operator is overloaded such that y += z adds z and y, then stores the result in y. Which of the following expressions is always equivalent to y += z?
   a. y.operator+=(z)
   b. y = y + z
   c. y = y operator+= z
   d. y operator+=(y + z)
13. For operators overloaded as non-static member functions:
   a. Both binary and unary operators take one argument.
   b. Neither binary nor unary operators can have arguments.
   c. Binary operators can have two arguments and unary operators can have one.
   d. Binary operators can have one argument, and unary operators cannot have any.
14. Which of the following is false?
   a. Two arrays cannot be meaningfully compared with equality or relational operators.
   b. C++ ensures that you cannot "walk off" either end of an array.
   c. An entire non-char array cannot be input or output at once.
   d. Arrays cannot be assigned to one another (i.e., array1 = array2;).
15. The array subscript operator [], when overloaded, cannot:
   a. Take multiple values inside (e.g., [4 8]).
   b. Take a float as an operand.
   c. Take user-defined objects as operands.
   d. Be used with linked list classes.
16. A copy constructor:
   a. is a constructor that takes no arguments.
   b. is a constructor with only default arguments.
   c. is a constructor that initializes a newly declared object to the value of an existing object of the same class.
   d. None of the above.
17. Copy constructors must receive its argument by reference because:
   a. the pointer needs to know the address of the original data, not a temporary copy of it.

b. the copy of the argument passed by value has function scope.
c. otherwise the constructor will only make a copy of a pointer to an object.
d. otherwise infinite recursion occurs.

18. To prevent class objects from being copied:
    a. Make the copy constructor private.
    b. Make the overloaded assignment operator private.
    c. Both (a) and (b).
    d. None of the above.
19. Conversion constructors:
    a. Can convert between user-defined types.
    b. Are implicitly defined by the compiler if not explicitly written by the programmer.
    c. Cannot convert built-in types to user defined types.
    d. Can have multiple arguments.
20. The prototypes of overloaded cast operator functions do not:
    a. Specify the type they convert to.
    b. Need to be defined inside the class whose objects are being converted.
    c. Specify the type that is being converted.
    d. Specify a return type.
21. Which of the following lines would be the prototype for an overloaded cast operator function that converts an object of user-defined type Time into a double?
    a. Time::operator_cast(double) const
    b. Time::static_cast double() const;
    c. Time::operator double() const;
    d. d. Time::double() const;.

True False

1. Using the operator overloading concept, we can change the meaning of an operator.
2. Operator overloading works when applied to class object.
3. Friend functions cannot be used to overload operators.
4. When using an overloaded binary operator, the left operand is implicitly passed to the member function.
5. The overloaded operator must have at least one operand that is user-defined type.
6. Trough operator overloading, a class type data can be converted to a basic type data.
7. Operator functions never return a value.
8. A constructor can be used to convert a basic type to a class type data.
9. Precedence and associativity of overloaded operators can be changed.
10. Semantics of overloaded operators can be changed.
11. With overloading binary operator, the left and right operands are explicitly passed.
12. The overloaded operator function parameters must be user-defined object only.
13. A constructor can be used to convert a user-defined data types only.
14. An object of a class can be assigned to basic type operand.
15. Syntax of overloaded operators can be changed.
16. The parameter type to overloaded subscript [] operator can be of any data type.
17. Friend function can access members of a class directly.

18. The ternary operator can be overloaded.
19. The compiler reports an error if overloaded + operator perform – operation.

Fill in the blanks

1. Operator overloading is called _____.
2. When overloading the ____ operator, it is important to consider if the programmer will want to/should be allowed to use this operator as an Lvalue (on the left side of an assignment).
3. The "==" operator compares for _____.
4. A bool value returns _____.
5. Overloading an operator allows operators to be used with a _____.
6. A function with 2 arguments is _____.
7. + and – are _____ operators.
8. += and -= are _____ operators.
9. A binary function is a function with _____ arguments.
10. Defining a new meaning for an operator is called _____.
11. The _____ operator can be used with a new class without defining this binary function.
12. =< is a _____ operator.
13. _____ allow the == operator to be used with a new class.
14. With overloading _____, the left and right operands are explicitly passed.
15. Trough operator overloading, a _____ data can be converted to a _____ data.
16. The array subscript operator [], when overloaded, cannot _____.
17. A _____ can be used to convert a user-defined data types only.
18. An _____ of a class can be assigned to basic type operand.

Inheritance

Unit – 4

Short questions

1. Differentiate between object based and object oriented programming.
2. What is difference between public and private inheritance?
3. What is difference between protected inheritance and other types of inheritance?
4. Define abstract class.
5. Define base class.
6. What is base constructor?
7. What is child class?
8. Define inheritance.
9. What is single inheritance?
10. What is multiple inheritance?
11. Define multilevel inheritance.
12. Define hybrid inheritance.
13. What is derived class?
14. What is virtual base class?

15. What is direct base class?
16. What is indirect base class?
17. What is derived class constructor?
18. Define nesting of classes.
19. Define grandparent class.
20. Define grandfather class.
21. What is containership or delegation?
22. Define common base class.
23. Define duplicate member.

Long Questions

1. What are the advantages of using Inheritance?
2. Explain the statement. "The private members of the base class are indirectly available to the derive class."
3. Explain how different types of data members are treated under different types of inheritance.
4. How is protected access specifier different from other access specifiers while further inheriting a class?
5. Why does the C++ object model implement the base class subjects in the derived class object?
6. What are the disadvantages of Multiple Inheritance?
7. What are the issues one must consider when dealing with multiple inheritance?
8. How are constructors of the derived classes executed? What are the issues one must consider while writing derive class constructor?
9. What does inheritance mean in C++? Explain with one example.
10. What are the different forms of inheritance? Give an example for each.
11. Describe the syntax of the single inheritance in C++.
12. We know that a private member of a base class is not inheritable. Is it any way possible for the objects of a derived class to access the private members of the base class? If yes, how? Remember, the base class cannot be modified.
13. How do the properties of the following two derived classes differ?
    a. Class D1:private B{//…};
    b. class D2:public B{//..};
14. When do we use the protected visibility specifier to a class member?
15. Describe the syntax of Multiple Inheritance. When do we use such an inheritance?
16. What are the implications of the following two definitions?
    a. Class A:public B, public C{//….};
    b. Class A:public C, public B{//…};
17. What is virtual base class? Explain with example.
18. When do we make class virtual?
19. What is an abstract class? Explain with example.
20. In what order are the class constructor called when a derived class object is created?
21. Class D is derived from class B. The class D does not contain any data members of its own. Does the class D require constructors? If yes, why?

22. What is containership? How it differs from inheritance?
23. Describe how an object of a class that contains objects of other classes created?
24. What are the differences between the access specifiers private and protected?
25. What are base and derived classes? Create a base class called stack and derived class called MyStack. Write a program to use these classes for manipulating objects.
26. Explain the syntax for declaring the derived class. Draw access privilege diagram for members of a base and derived class.
27. What are the difference between a C++ struct and C++ class in terms of encapsulation and inheritance?
28. What are the different forms of inheritance supported by C++? Explain them with example.
29. What is class hierarchy? Explain how inheritance helps in building class hierarchy.
30. Can base class, access members of a derived class? Give reason.
31. What is visibility mode? What are the different inheritance visibility modes supported by C++?
32. What is the difference between inheriting a class with a public and private visibility mode?
33. Explain how base class member function can be invoked in a derived class if the derived class also has a member function with the same name.
34. Consider an example of declaring the examination result. Design three classes: Student, Exam and Result. The Student class has data members such as those representing roll number, name etc. create a class Exam by inheriting the student class. The exam class adds data members representing the marks scored in six subjects. Derive class Result from the exam class and it has own data members such as total_marks. Write an interactive program to model this relationship. What type of inheritance this model belongs to?
35. Discuss cost and benefits of inheritance emphasizing ease of design, code reusability, over head etc.

Multiple Choice Questions

1. Which allows you to create a derived class that inherits properties from more than one base class?
   a. Multilevel inheritance
   b. Multiple inheritance
   c. Hybrid Inheritance
   d. Hierarchical Inheritance
2. Which feature in OOP allows reusing code?
   a. Polymorphism
   b. Inheritance
   c. Encapsulation
   d. Data hiding
3. Which of the following members do get inherited but become private members in child class
   a. Public
   b. Private

c. Protected
d. All of above

4. class derived: public base1, public base2 { } is an example of
   a. Polymorphic inheritance
   b. Multilevel inheritance
   c. Hierarchical inheritance
   d. Multiple inheritance
5. A class defined within another class is:
   a. Nested class
   b. Inheritance
   c. Containership
   d. Encapsulation
6. The major goal of Inheritance in c++ is:
   a. to facilitate the conversion of data types.
   b. to help modular programming.
   c. to extend the capabilities of a class.
   d. to hide the details of base class.
7. Consider the following class definition

   ```
   Class a
   {
   };
   Class B:protected a
   {
   };
   ```

   What happens when we try to compile this class?
   a. Will not compile because class body of a is not defined.
   b. Will not compile because class body of b is not defined.
   c. Will not compile because class a is not public inherited.
   d. Will compile successfully.
8. Advantages of inheritance include
   a. providing class growth through natural selection
   b. facilitating class libraries
   c. avoiding the rewriting of code
   d. none of the above
9. A class hierarchy (inheritance)
   a. shows the same relationships as an organization chart
   b. describes "has a" relationships
   c. describes "is a kind of" relationships
   d. shows the same relationships as a family tree
10. Virtual functions allow you to:
    a. create an array of type pointer-to-base-class that can hold pointers to derived classes
    b. create functions that have no body
    c. group objects of different classes so they can all be accessed by the same function code

d. use the same function call to execute member functions of objects from different classes

11. A pure virtual function is a function that:
    a. has no body
    b. returns nothing
    c. is used in a base class
    d. takes no argument
12. Abstract class cannot have _____.
    a. Zero instance
    b. Multiple instance
    c. Both zero instance & multiple instance.
    d. None of these option.
13. Which is a logical abstract class for a class called "CricketPlayer"
    a. Bank
    b. Athlete
    c. Sport
    d. Team
14. _____ is a relationship.
    a. Polymorphism
    b. Inheritance
    c. Overloading
    d. None of these option
15. A derived class _____.
    a. Inherits data members and member functions from base class.
    b. Inherits constructors and destructors.
    c. Object can access protected members with the dot operator.
    d. None of the above.
16. Can two classes contain member functions with the same name?
    a. No
    b. Yes, but only if the two classes have the same name.
    c. Yes, but only if the main program does not declare both kinds
    d. Yes, this is always allowed.
17. Which is a logical abstract base class for a class called "footballplayer".
    a. Salary
    b. Sport
    c. Athlete
    d. Team

True False

1. Inheritance helps in making a general class into a more specific class.
2. Inheritance aids data hiding.
3. One of the advantages of inheritance is that it provides a conceptual framework.
4. Inheritance facilitates the creation of class libraries.
5. Defining a derived class requires some changes in the base class.
6. A base class is never used to create objects.

7. It's a legal to have an object of one class as a member of another class.
8. We can prevent the inheritance of all members of the base class by making base class virtual in the definition of the derived class.
9. Both base and derived classes need not have constructors.
10. Only base class cannot have constructors.
11. Only derived class can have constructors.
12. No-argument constructor of the base class is invoked when a derived class is instantiated.
13. Derived class member cannot access private members of a base class.
14. When a derived class is instantiated only the derived class constructors are invoked.
15. When derive class is instantiated, memory is allocated to all data members of both the base and derive class.
16. Constructor is invoked starting from the top base class to derived class order.
17. Destructor is invoked in the reverse order of constructor.
18. Destructor is invoked starting from the top base class to derive class order.
19. Base class constructors can be explicitly invoked in the reverse order of constructor.
20. If a base class does not have no-argument constructor and has parameterized constructors, must be explicitly invoked from a derived class.

Fill in the blanks

1. Abstract class cannot have _____.
2. _____is the capability of one class to inherit properties from another class.
3. _____is the class whose properties are inherited by another class. It is also called Super Class.
4. _____is the class that inherit properties from base class(es).It is also called Sub Class.
5. _____ is the inheritance hierarchy wherein one derived class inherits from one base class.
6. _____ is the inheritance hierarchy wherein one derived class inherits from multiple base class(es).
7. _____ is the inheritance hierarchy wherein multiple subclasses inherit from one base class.
8. _____ is the inheritance hierarchy wherein subclass acts as a base class for other classes.
9. The inheritance hierarchy that reflects any legal combination of other four types of inheritance is called _____.
10. It is the keyword that controls the visibility and availability of inherited base class members in the derived class called _____.
11. It is the inheritance facilitated by private visibility mode known as _____.
12. When a class contains objects of other class types as its members, it is called _____.
13. When both derived and base class contains constructors, the _____ constructor is executed first and then the constructor in the _____ class is executed.
14. Multipath inheritance may lead to duplication of inherited members from a grandparent base class. This may be avoided by _____.

15. _____ is a relationship.
16. A derived class _____.
17. One of the advantages of inheritance is that it provides a _____.
18. Inheritance aids _____.
19. No-argument constructor of the base class is invoked when a _____ is instantiated.
20. Derived class member cannot access private members of a _____.

Polymorphism

Unit – 5

Short Question

1. Define 'address of' operator.
2. What is abstract base class?
3. What is compile time polymorphism?
4. What is dynamic binding?
5. What is function overloading?
6. What is indirection operator?
7. What is late binding?
8. What is pointer?
9. What is polymorphism?
10. Define pure virtual function?
11. What is run time polymorphism?
12. What is static binding?
13. What is static linking?
14. Example of virtual constructor.
15. Define virtual constructor.
16. Example of virtual destructor.
17. Define virtual destructor.
18. What is virtual function?
19. Types of polymorphism.
20. Types of compile time polymorphism.

Long Questions

1. What is polymorphism? What is the difference between compile time and runtime polymorphism?
2. What is the importance of this pointer in the call to the function using base class pointer?
3. Consider base class B and derive class D. Assume pB is a pointer to base class and pD is pointer to derived class. Differentiate between these two pointers in terms of accessing the derived class object.
4. Why is a non-virtual member function of a base class always called even when the base class pointer is pointing to the derived class object?
5. What is the difference between a normal member function and a virtual function?

6. Explain static binding. How do virtual functions enable dynamic binding?
7. What is the problem of default arguments of a virtual function? How can that be resolved?
8. Differentiate between virtual function and pure virtual functions.
9. What does polymorphism mean in C++ language?
10. How is polymorphism achieved at compile time, and run time?
11. Discuss different ways by which we can access public member functions of an object.
12. Explain, with an example, how you would create space for an array of objects using pointer?
13. What does **this** pointer point to?
14. What are the applications of **this** pointer?
15. What is virtual function? Explain with example.
16. Why do we need virtual function?
17. When do we make a virtual function "**pure**"?
18. What are the implications of making a function a pure virtual function?
19. Describe different methods of realizing polymorphism in C++
20. Justify the need for virtual functions in C++.
21. Why C++ supports type compatibles pointers unlike C?
22. What is run time dispatching? Explain how C++ handles run time dispatching.
23. What are pure virtual functions? How they differ from normal virtual function?
24. What are abstract classes? Explain with example.
25. What is virtual destructor? Explain with example.
26. How do virtual destructors differ from normal destructors?
27. Can constructors be declared as virtual constructors? Give reason.
28. Explain how dynamic binding is achieved by the C++ compilers?
29. What are the rules that need to be kept in mind in deciding virtual functions?

Multiple Choice Questions

1. _____ means 'one name, multiple forms'.
   a. Polymorphism
   b. Inheritance
   c. Encapsulation
   d. None of above
2. Concept of polymorphism is implemented using
   a. Function overloading
   b. Operator overloading
   c. A and B
   d. A only
3. An object is bound to its function call at compile time known as
   a. Early binding
   b. Compile time polymorphism
   c. Static binding
   d. All of above
4. How can we achieve run time polymorphism in C++?
   a. Friend Function

b. Virtual function
c. Operator overloading
d. Function overloading

5. Polymorphism is divided into _____ types.
   a. Two
   b. Three
   c. Four
   d. None of above
6. _____ is a derived data type that refers to another data variable by storing the variable's memory address rather than data.
   a. Pointers
   b. Polymorphism
   c. Structure
   d. Inheritance.
7. The function is linked with a particular class much later after the compilation, this process is termed as
   a. Static binding
   b. Late binding
   c. Static linking
   d. None of above
8. A pointer can be incremented and decremented by
   a. ++
   b. –
   c. A and B
   d. A only
9. When we use the same function name in both the base and derived classes, the function in base class is declared as _____
   a. Friend
   b. Virtual
   c. A and b
   d. None of above
10. Which of the following is not a rule for virtual functions
    a. The virtual function must be members of some class.
    b. They cannot be static members.
    c. They are accessed by using object pointer
    d. None of above
11. _____ is a function declared in a base class that has no definition relative to the base class.
    a. Pure virtual function
    b. Virtual function
    c. Friend function
    d. None of above
12. Late binding and early binding is a part of _____
    a. Binding
    b. Dynamic binding

c. Polymorphism
d. None of above

13. A _____ pointer refers to an object that is currently invokes a member function.
    a. this
    b. these
    c. that
    d. none of above
14. We can have virtual destructor but not have constructor.
    a. True
    b. False
    c. May be
    d. None of above
15. A virtual function, equated to zero is called _____
    a. Pure virtual function
    b. Virtual function
    c. Friend function
    d. None of above

True False

1. Virtual functions are used to create pointers to base classes.
2. Virtual function allows us to use the same functions call to invoke member functions of objects of different classes.
3. A pointer to a base class cannot be made to point to objects of derived class.
4. **this** pointer points to the objects that is currently used to invoke a function.
5. **this** pointer can be used like any other pointer to access the members of the object it points to.
6. **this** pointer can be made to point to any object by assigning the address of the object.
7. Pure virtual functions force the programmer to redefine the virtual function inside the derived classes.
8. In C++, pointers to int data type can be used to point to float types.
9. Pointer to base class can point to an object of any class.
10. Pointer to a class at the top of the class hierarchy can point to any class objects in that hierarchy.
11. Virtual function allows invoking different function with the same statement.
12. The sizeof a class having virtual function is the same as that without virtual function.
13. A class with virtual function can be instantiated.
14. A class with pure virtual function can be instantiated.
15. A class with a pure virtual function is created by designer whereas, derived classes are created by programmers.
16. Specification of a virtual function in the base class and its derived class must be same.
17. Pure virtual functions postpone implementation of a member function to its derived class.

Fill in the blanks

1. _____functions postpone implementation of a member function to its derived class.
2. Specification of a virtual function in the _____ and its derived class must be same.
3. _____ allows to invoke different function with the same statement.
4. The _____ a class having virtual function is the same as that without virtual function.
5. _____ pointer can be made to point to any object by assigning the address of the object.
6. _____ pointer can be used like any other pointer to access the members of the object it points to.
7. Virtual functions are used to create _____ to _____.
8. Pure virtual functions force the programmer to redefine the _____ inside the _____ classes.
9. A pointer to a base class cannot be made to point to objects of _____ class.
10. Two or more functions may have the same name, as long as their _____ are different.
11. When writing function or class template, one use a _____ to specify a generic data type.
12. Run time polymorphism is_____ than the compile time polymorphism.
13. The two types of polymorphism is : _____ & _____.
14. _____ are blue prints of a function that can be applied to different data types.
15. We can convert a class data type to basic data type using _____.
16. Function overloading is an example of _____.
17. In _____ all data elements are stored in the same memory location.

Streams I/O Operations

Unit – 6

Short Questions

1. What is stream?
2. What is use of fill()?
3. What is use of get()?
4. State use of getline().
5. State use of precision() function.
6. State different stream classes.
7. Which class is the base class for the input stream and output stream?
8. State use of ios class.
9. Which type of class ios is?
10. Which classes are the base class for the iostream?
11. List different type of get() function.
12. Define use of streambuf class.
13. List methods provided by istream and ostream.
14. Write syntax of put().
15. Which function is used to read and display line of text?

16. List methods that are used for formatting the output.
17. Which function is used to set precision?
18. Which function is used to fill unused positions by any desire character?
19. Setf() function belong to which class?
20. How long the flags set by setf() remain effective?

Long Questions

1. What is the difference between IO provided by C and C++?
2. What is the importance of ios member flags in formatting IO?
3. How can we clear all the flags for formatting at the same time?
4. What are the difference between manipulators and ios functions?
5. What is stream? Explain the features of C++ stream I/O with C's I/O system.
6. Describe briefly the features of I/O system supported by C++.
7. How do the I/O facilities in C++ differ from that in C?
8. Why are the words such as **cin** and **cout** not considered as keywords?
9. How is cout able to display various types of data without any special instruction?
10. Why it is necessary to include the file iostream in all program?
11. Discuss the various form of get() function supported by the input stream. How are they used?
12. How do the following two statements differ in operation?
    Cin>> c;
    Cin.get(c);
13. Both cin and getline() function can be used for reading a string. Comment.
14. Discuss the implications of size parameter in the following statement:
    cout.write(line,size);
15. What does the following statement do?
    cout.write(s1,m).write(s2,n);
16. What role does the iomainp file play?
17. What is the role of file() function? When do we use this function?
18. Discuss the syntax of set() function.
19. What is the basic difference between manipulators and **ios**member function in implementation? Give example.
20. List C++ predefined streams and explain them with suitable example program.
21. Draw console stream class hierarchy and explain its members.
22. What is the difference between the statements?
    cin>>ch;
    ch=cin.get();
23. What is the output of the following statements:
    a. cout<< 65;
    b. cout.put(65);
    c. cout.put('A');
24. Explain the various methods of performing formatted stream I/O operations.
25. How are the input and output streams tied using istream.tie() member function?
26. What are the custom manipulators?

Multiple Choice Questions

1. _____ is a sequence of bytes.
   a. Stream
   b. Input stream
   c. Output stream
   d. None of above
2. The source stream that provides data to the program is called
   a. Stream
   b. Input stream
   c. Output stream
   d. None of above
3. The destination stream that receives output from the program is called
   a. Stream
   b. Input stream
   c. Output stream
   d. None of above
4. The data in the input stream can come from the
   a. Keyboard
   b. Monitor
   c. A and b
   d. None of above
5. Which class is the base class for rest of the classes in stream classes?
   a. ios
   b. istream
   c. streambuf
   d. none of above
6. The class ios declared as
   a. Pure virtual base class
   b. Virtual base class
   c. Abstract class
   d. None of above
7. _____ class provides the basic support for formatted and unformatted I/O operations.
   a. ios
   b. istream
   c. streambuf
   d. ostream
8. The class _____ provides the facilities for formatted and unformatted input.
   a. istream
   b. istream_withassign
   c. iostream
   d. streambuf
9. The class _____ provides the facilities for formatted and unformatted output.
   e. ostream

f. ostream_withassign
g. iostream_withassign
h. streambuf

10. istream declares input functions such as
    a. get()
    b. getLine()
    c. read()
    d. all of above
11. ostream declares output functions such as
    a. put()
    b. write()
    c. a and b
    d. none of above
12. iostream inherits the properties of
    a. ios
    b. istream
    c. ostream
    d. all of above
13. _____ provides an interface to physical devices through buffers.
    a. BufferReader
    b. Streambuff
    c. Streambuf
    d. None of above
14. istream class and << is overloaded in the _____ class
    a. istream
    b. ostream
    c. ios
    d. none of above
15. The operator _____ reads the data character by character and assigns it to the indicated location.
    a. >>
    b. <<
    c. << >>
    d. None of above
16. Which of the following is not a part of ios formatted function?
    a. width()
    b. fill()
    c. setf()
    d. height()
17. The header file _____ provides a set of manipulator functions to manipulate output formats.
    a. stdio
    b. iomanip
    c. A and B
    d. None of above

True False

1. A C++ is a stream file.
2. C++ never truncates data.
3. The main advantage of width() function is that we can use one width specification for more than one items.
4. The get(void) functions provides a single-character input that does not skip over the white space.
5. The header file iomainp can be used in place of iostream.
6. We cannot use both the C I/O functions and C++ I/O functions in the same program.
7. A programmer can define a manipulator that could represent a set of format functions.
8. The << operator inserts the data that follows it into the stream preceding it.
9. A stream is a sequence of bytes and serves as a source for an I/O data.
10. C++ stream classes are declared in the header file "iostream".
11. cout represent the input stream connected to the standard input device.
12. Cin represents the output stream connected to the standard output device.
13. The istream class define one member function put() to handle single character.
14. The ostream class define one member function get() to handle single character.
15. The >> operator is overloaded in the istream class.
16. We can read line of text using the line oriented I/O function getline().
17. We can write line of text using the line oriented I/O function write().
18. width(), precision(), fill(), setf() and unsetf() functions are belong to ios class.
19. The header file iomainp provides a set of manipulator functions to manipulate output formats.

Fill in the blanks

1. The _____ function provides a single-character input that does not skip over the white space.
2. The main advantage of _____ function is that we can use one width specification for more than one item.
3. The header file _____ can be used in place of iostream.
4. A programmer can define a _____ that could represent a set of format functions.
5. The _____ operator inserts the data that follows it into the stream preceding it.
6. The _____ operator read the data that follows it into the stream preceding it.
7. A _____ is a sequence of bytes and serves as a source for an I/O data.
8. C++ stream classes are declared in the header file _____.
9. _____ represent the input stream connected to the standard input device.
10. _____ represent the output stream connected to the standard output device.
11. The istream class defines one member function _____to handle single character.
12. The ostream class defines one member function _____ to handle single character.

13. The >> operator is overloaded in the _____ class.
14. We can read line of text using the line oriented I/O function _____.
15. We can write line of text using the line oriented I/O function _____.
16. width(), precision(), fill(), setf() and unsetf() functions are belong to _____ class.
17. The header file _____ provides a set of manipulator functions to manipulate output formats.