2 Marks and 16 Marks

Class : III EEE

Subject Code : CS2311

Subject : Object Oriented Programming

Part - A

1) Give the evolution diagram of OOPS concept.

Machine language

Procedure language

Assembly language

OOPS

2) Define Recursion?

Recursion is the process of defining something in terms of itself.Recrsion is the attribute that allows a method to call itself, A method that call itself is said to be recursive.

```
int fact(int n){

int result;

if (n= =1) return 1;

result= fact(n-1) * n;

return result;

}
```

3) What is Procedure oriented language?

Conventional programming, using high-level language such as COBOL, FORTRAN and C are commonly known as Procedure oriented language (POP). In POP number of functions are written to accomplish the tasks such as reading, calculating and printing.

4) Give some characteristics of procedure-oriented language.

_ Emphasis is on doing things (algorithms).

_ Larger programs are divided into smaller programs known as functions.

_ Most of the functions share global data.

_ Data move openly around the system from function to function.

_ Employs *top-down* approach in program design.

Function-1 Function-2 Function-3

Function-4 Function-5

Function-6 Function-7 Function-8

Main program

5) Write any four features of OOPS.

_ Emphasis is on data rather than on procedure.

_ Programs are divided into objects.

_ Data is hidden and cannot be accessed by external functions.

_ Follows **bottom-***up* approach in program design.

6) What are the basic concepts of OOS?

_ Objects.

_ Classes.

_ Data abstraction and Encapsulation.

_ Inheritance.

_ Polymorphism.

_ Dynamic binding.

_ Message passing.

7) What are objects?

Objects are basic run-time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle. Each object has the data and code to manipulate the data and theses objects interact with each other.

8)What is a class?

The entire set of data and code of an object can be made a user-defined data type with the help of a class. Once a class has been defined, we can create any number of objects belonging to the classes. Classes are user-defined data types and behave like built-in types of the programming language.

9) What is encapsulation?

Wrapping up of data and function within the structure is called as encapsulation.

10)What is data abstraction?

The insulation of data from direct access by the program is called as data hiding or information binding. The data is not accessible to the outside world and only those functions, which are wrapped in the class, can access it.

11)What are data members and member functions?

Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight, and cost and uses functions to operate on these attributes. The attributes are sometimes called as data members because they hold information. The functions that operate on these data are called as methods or member functions.

Eg: int a,b; // a,b are data members

Void getdata ( ) ; // member function

12)What is dynamic binding or late binding?

Binding refers to the linking of a procedure to the code to be executed in response to the call. Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at the run-time.

13)Write the process of programming in an object-oriented language?

_ Create classes that define objects and their behavior.

_ Creating objects from class definition.

_ Establishing communication among objects.

14)Give any four advantages of OOPS.

- The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
- It is possible to have multiple instances of an object to co-exist without any interference.
- Object oriented programming can be easily upgraded from small to large systems.
- Software complexity can be easily managed.

15)What are the features required for object-based programming Language?

_ Data encapsulation.

_ Data hiding and access mechanisms.

_ Automatic initialization and clear up of objects.

_ Operator overloading.

16)What are the features required for object oriented language?

_ Data encapsulation.

_ Data hiding and access mechanisms.

_ Automatic initialization and clear up of objects.

_ Operator overloading.

_ Inheritance.

_ Dynamic binding.

17)Give any four applications of OOPS

• Real-time systems.

• Simulation and modeling.

• Object-oriented databases.

• AI and expert systems.

18) Give any four applications of c++?

_ Since c++ allows us to create hierarchy-related objects, we can build special object-oriented libraries, which can be used later by many programmers.

_ C++ are easily maintainable and expandable.

_ C part of C++ gives the language the ability to get close to the machine-level details.

_ It is expected that C++ will replace C as a general-purpose language in the near future.

19) What are tokens?

The smallest individual units in a program are known as tokens. C++ has the following tokens,

o Keyword

o Identifiers

o Constants

o Strings

o Operator

20)What are keywords?

The keywords implement specific C++ language features. They are explicitly reserved identifiers and cannot be used as names fro the program variables or other user defined program elements.

Eg: go to, If, struct , else ,union etc.

21) Rules for naming the identifiers in C++.

_ Only alphabetic characters, digits and underscore are permitted.

_ The name cannot start with a digit.

_ The upper case and lower case letters are distinct.

_ A declared keyword cannot be used as a variable name.

22)What are the operators available in C++?

All operators in C are also used in C++. In addition to insertion operator << and extraction operator >> the other new operators in C++ are,

: Scope resolution operator

: : * Pointer-to-member declarator

->* Pointer-to-member operator

.* Pointer-to-member operator

delete Memory release operator

endl Line feed operator

new Memory allocation operator

setw Field width operator

23)What is a scope resolution operator?

Scope resolution operator is used to uncover the hidden variables. It also allows access to global version of variables.

Eg:

```
#include<iostream.h>

int m=10; // global variable m

void main ( )

{

int m=20; // local variable m

cout<<"m="<<m<<"\\n";

cout<<": : m="<<: : m<<"\\n";

}
```

output:

20

10 (: : m access global m)

Scope resolution operator is used to define the function outside the class.

Syntax:

Return type <class name> : : <function name>

Eg:

Void x : : getdata()

24) What are free store operators (or) Memory management operators?

New and Delete operators are called as free store operators since they allocate the

memory dynamically.

New operator can be used to create objects of any data type.

Pointer-variable = new data type;

Initialization of the memory using new operator can be done. This can be done as,

Pointer-variable = new data-type(value)

Delete operator is used to release the memory space for reuse. The general form of

its use is

Delete pointer-variable;

25) What are manipulators?

Setw, endl are known as manipulators.

Manipulators are operators that are used to format the display. The endl manipulator

when used in an output statement causes a linefeed to be inserted and its effect is similar

to that of the newline character"\\n".

Eg: Cout<<setw(5)<<sum<<endl;

26) What do you mean by enumerated datatype?

An enumerated datatype is another user-defined datatype, which provides a way for attaching names to numbers, thereby increasing comprehensibility of the code.

The syntax of an **enum** statement is similar to that of the struct statesmen.

Eg:

enum shape{ circle, square, triangle}

enum color{ red, blue, green, yellow}

27) What are symbolic constants?

There are two ways for creating symbolic constants in C++:

_ Using the qualifier **constant.**

_ Defining a set of integer constants using **enum** keyword.

The program in any way cannot modify the value declared as constant in c++.

Eg: Const int size =10;

Char name [size];

28)What do you mean by dynamic initialization of variables?

C++ permits initialization of the variables at run-time. This is referred to as dynamic initialization of variables.

In C++ ,a variable can be initialized at run-time using expressions at the place

of declaration as,

……..

…....

int n =strlen(string);

……..

float area=3.14*rad*rad;

Thus declaration and initialization is done simultaneously at the place where the variable

is used for the first time.

29) What are reference variable?

A reference variable provides an alias(alternative name) for a previously defined variable.

For example , if make the variable **sum** a reference to the variable **total**, then

sum and **total** can be used interchancheably to represent that variable.

Syntax:

Data-type &reference-name = variable-name

Eg:

float total = 100;

float sum = total;

30)What is member-dereferencing operator?

C++ permits to access the class members through pointers. It provides three pointer-to-member operators for this purpose,

: :* To declare a pointer to a member of a class.

* To access a member using object name and a pointer to the member

->* To access a member using a pointer to the object and a pointer to that member.

31)what is function prototype ?

The function prototype describes function interface to the compiler by giving details

such as number ,type of arguments and type of return values Function prototype is a declaration statement in the calling program and is of the following

Type function_name(argument list);

Eg float volume(int x,float y);

32)what is an inline function ?

An inline function is a function that is expanded in line when it is invoked. That is compiler replaces the function call with the corresponding function code.

The inline functions are defined as

Inline function-header

{

function body

}

33) Write some situations where inline expansion may not work

_ for functions returning values, if loop, a **switch,** or a **goto** exists

_ for functions not returning values ,if a return statement exists

_ if function contain **static** variables

_ if **inline** functions are recursive

34)what is a default argument ?

Default arguments assign a default value to the parameter, which does not have matching argument in the function call. Default values are specified when the function is declared.

Eg : float amount(float principle,int period,float rate=0.15)

Function call is

Value=amount(5000,7);

Here it takes principle=5000& period=7

And default value for rate=0.15

Value=amount(5000,7,0.34)

Passes an explicit value 0f 0.34 to rate

We must add default value from right to left

35) What are constant arguments ?

keyword is const. The qualifier const tells the compiler that the function should not modify the argument. The compiler will generate an error when this condition is violated. This type of declaration is significant only when we pass arguments by reference or pointers

eg: int strlen(**const** char *p);

36) How the class is specified ?

Generally class specification has two parts

_ class declaration

It describes the type and scope of its member

_ class function definition

It describes how the class functions are implemented

The general form is

Class class_name

{

private:

variable declarations;

function declaration;

public:

variable declaration;

function declaration;

};

37) How to create an object ?

Once the class has been declared, we can create variables of that type by using the classname

Eg:classname x; //memory for x is created

38) How to access a class member ?

object-name. function-name(actual arguments)

eg:x.getdata(100,75.5);

39) How the member functions are defined ?

Member functions can be defined in two ways

_ outside the class definition

Member function can be defined by using scope resolution operator::

General format is

Return type **class**_name::function-name(argument declaration)

{

}

_ Inside the class definition

This method of defining member function is to replace the function

declaration by the actual function definition inside the class. It is treated as inline

function

Eg:class item

{

int a,b ;

void getdata(int x,int y)

{

a=x;

b=y;

};

40) What is static data member?

Static variable are normally used to maintain values common to the entire class.

Feature:

_ It is initialized to zero when the first object is created. No other initialization is

permitted

_ only one copy of that member is created for the entire class and is shared by all

the objects

_ It is only visible within the class, but its life time is the entire class

_ type and scope of each static member variable must be defined outside the class

_ It is stored separately rather than objects

Eg: static int count//count is initialized to zero when an object is created.

int classname::count;//definition of static data member

41) What is static member function?

A member function that is declared as static has the following properties

_ A static function can have access to only other static member declared in the

same class

_ A static member function can be called using the classname as follows

classname ::function_name;

42) How the objects are used as function argument?

This can be done in two ways

_ A copy of the entire object is passed to the argument

_ Only address of the objects is transferred to the function

43) What is called pass by reference?

In this method address of an object is passed, the called function works directly on the actual arguments.

44) Define const member

If a member function does not alter any data in the class, then we may declare it as const member function as

Void mul(int ,int)const;

45) Define pointers to member

It is possible to take the address of a member of a class and assign it to a pointer. The address of a member can be obtained by applying the operator &to a “fully qualified” class member name. A class member pointer can be declared using the operator::*with the class name.

Eg: class A

{

int m;

public:

void show( );

};

pointer to member m is defined as

int A::*ip=&A::m;

A::*->pointer to member of A class

&A::m->address of the m member of A class

46) When the deferencing operator ->* is used?

It is used to access a member when we use pointer to both the object and the member.

47) When the deferencing operator .* is used?

It is used to access a member when the object itself is used as pointers.

48) Define local classes.

Classes can be defined and used inside a function or a block. such classes are called local classes. It can use global variables and static variables declared inside the function but cannot use automatic local variables.

Eg;

void test(int a)

{

…….

}

class student

{

……

};

student s1(a);

}

49) Define constructor

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

Eg:

Class **integer**

{

……

public:

integer( );//constructor

……

}

50) Define default constructor

The constructor with no arguments is called default constructor

Eg:

Class integer

{

int m,n;

Public:

Integer( );

…….

};

integer::integer( )//default constructor

{

m=0;n=0;

}

the statement

integer a;

invokes the default constructor

51) Define parameterized constructor

constructor with arguments is called parameterized constructor

Eg;

Class integer

{ int m,n;

public:

integer(int x,int y)

{ m=x;n=y;

}

To invoke parameterized constructor we must pass the initial values as arguments to the constructor function when an object is declared. This is done in two ways

1.By calling the constructor explicitly

eg:

integer int1=integer(10,10);

2.By calling the constructor implicitly

eg:

Integer int1(10,10);

52) Define default argument constructor

The constructor with default arguments are called default argument constructor

Eg:

Complex(float real,float imag=0);

The default value of the argument imag is 0

The statement

complex a(6.0)

assign real=6.0 and imag=0

the statement

complex a(2.3,9.0)

assign real=2.3 and imag=9.0

53) What is the ambiguity between default constructor and default argument

constructor ?

The default argument constructor can be called with either one argument or no arguments. when called with no arguments ,it becomes a default constructor. When both these forms are used in a class ,it cause ambiguity for a statement such as A a; The ambiguity is whether to call A::A() or A::A(int i=0)

54) Define copy constructor

A copy constructor is used to declare and initialize an object from another object. It takes a reference to an object of the same class as an argument

Eg: integer i2(i1);

would define the object i2 at the same time initialize it to the values of i1.

Another form of this statement is

Eg: integer i2=i1;

The process of initializing through a copy constructor is known as **copy initialization**.

55) Define dynamic constructor

Allocation of memory to objects at time of their construction is known as dynamic constructor. The memory is allocated with the help of the **NEW operator**

Eg:

```
Class string

{

char *name;

int length;

public:

string( )

{

length=0;

name=new char[length +1];

}

void main( )
```

{

string name1(“Louis”),name3(Lagrange);

}

56) Define const object

We can create constant object by using const keyword before object declaration.

Eg: Const matrix x(m,n);

57) Define destructor

It is used to destroy the objects that have been created by constructor. Destructor name is same as class name preceded by tilde symbol(~)

Eg;

~integer()

{

}

A destructor never takes any arguments nor it does it return any value. The compiler upon exit from the program will invoke it.

Whenever **new** operator is used to allocate memory in the constructor, we should use delete to free that memory.

58) Define multiple constructors (constructor overloading).

The class that has different types of constructor is called multiple constructors

Eg:

#include<iostream.h>

#include<conio.h>

```
class integer
{
int m,n;
public:
integer( ) //default constructor
{
m=0;n=0;
}
integer(int a,int b) //parameterized constructor
{
m=a; n=b;
}
integer(&i) //copy constructor
{
m=i.m;
n=i.n;
}
void main()
{
integer i1; //invokes default constructor
integer i2(45,67);//invokes parameterized constructor
integer i3(i2); //invokes copy constructor
}
```

59) Write some special characteristics of constructor

• **T**hey should be declared in the public section

• They are invoked automatically when the objects are created

• They do not have return types, not even void and therefore, and they cannot return values

• They cannot be inherited, though a derived class can call the base class

• They can have default arguments

• Constructors cannot be virtual function

60) How the objects are initialized dynamically?

To call parameterized constructor we should the pass values to the object

ie,for the constructor integer(int a,int b)

it is invoked by integer a(10,18)

this value can be get during run time. i.e., for above constructor

int p,q;

cin>>p>>q;

integer a(p,q);

61)Define Inline Function?

Inline function is defined as a function definition such that each call to the function is in effect, replaced by the statements that define the function. It is expanded in line when it is invoked. The general form is

inline function-header

{

function body

}

62)Explain return by reference with an example.

A function can also return a reference. Consider the following function

int & max(int &x , int &y)

{ if(x>y)

return x;

else

return y;

}

Since the return type of max ( ) is int & the function returns reference to x or y (and not the values). Then a function call such as max (a , b) will yield a reference to either a or b depending on their values.

The statement

max (a , b) = -1;

is legal and assigns –1 to a if it is larger, otherwise –1 to b.

63) What are Friend functions? Write the syntax

A function that has access to the private member of the class but is not itself a member of the class is called friend functions.

The general form is

friend data_type function_name( );

Friend function is preceded by the keyword ‘friend’.

64)Write some properties of friend functions.

_ Friend function is not in the scope of the class to which it has been declared as friend. Hence it cannot be called using the object of that class.

_ Usually it has object as arguments.

_ It can be declared either in the public or private part of a class.

_ It cannot access member names directly. It has to use an object name and

dot membership operator with each member name. eg: (A . x)

65)What are virtual functions?

A function qualified by the 'virtual' keyword is called virtual function. When a virtual function is called through a pointer, class of the object pointed to determine which function definition will be used.

66)Write some of the basic rules for virtual functions

_ Virtual functions must be member of some class.

_ They cannot be static members and they are accessed by using object pointers

_ Virtual function in a base class must be defined.

_ Prototypes of base class version of a virtual function and all the derived class

versions must be identical.

_ If a virtual function is defined in the base class, it need not be redefined in the

derived class.

67) What are pure virtual functions? Write the syntax.

A pure virtual function is a function declared in a base class that has no definition relative to the base class. In such cases, the compiler requires each derived class to either define the function or

redeclare it as a pure virtual function. A class containing pure virtual functions cannot be used to declare any object of its own. It is also known as “donothing” function.

The “do-nothing” function is defined as follows:

virtual void display ( ) =0;

68) What is polymorphism? What are its types?

Polymorphism is the ability to take more than one form. An operation may exhibit different behaviors in different. The behavior depends upon the type of data used.Polymorphism is of two types. They are

_ Function overloading

_ Operator overloading

69) What is function overloading? Give an example.

Function overloading means we can use the same function name to create functions that perform a variety of different tasks.

Eg: An overloaded add ( ) function handles different data types as shown below. // Declarations

i. int add(int a, int b); //add function with 2 arguments of same type

ii. int add(int a, int b, int c); //add function with 3 arguments of same type

iii. double add(int p, double q); //add function with 2 arguments of

different type

//Function calls

add (3 , 4); //uses prototype (i.)

add (3, 4, 5); //uses prototype (ii.)

add (3 , 10.0); //uses prototype (iii.)

70) What is operator overloading?

C++ has the ability to provide the operators with a special meaning for a data type.This mechanism of giving such special meanings to an operator is known as Operator overloading. It provides a flexible option for the creation of new definitions for C++ operators.

71) List out the operators that cannot be overloaded.

_ Class member access operator (. , .*)

_ Scope resolution operator (::)

_ Size operator (sizeof)

_ Conditional operator (?:)

72) What is the purpose of using operator function? Write its syntax.

To define an additional task to an operator, we must specify what it means in relation to the class to which the operator is applied. This is done by Operator function ,which describes the task. Operator functions are either member functions or friendfunctions. The general form is

return type classname :: operator (op-arglist)

{

function body

}

where *return type* is the type of value returned by specified operation.

Op-operator being overloaded. The *op* is preceded by a keyword operator. operator op is

the function name.

73) Write at least four rules for Operator overloading.

_ Only the existing operators can be overloaded.

_ The overloaded operator must have at least one operand that is of user

defined data type.

_ The basic meaning of the operator should not be changed.

_ Overloaded operators follow the syntax rules of the original operators.

They cannot be overridden.

74) How will you overload Unary & Binary operator using member functions?

When unary operators are overloaded using member functions it takes no explicit arguments and return no explicit values.

When binary operators are overloaded using member functions, it takes one explicit argument. Also the left hand side operand must be an object of the relevant class.

75) How will you overload Unary and Binary operator using Friend functions?

When unary operators are overloaded using friend function, it takes one reference argument (object of the relevant class) When binary operators are overloaded using friend function, it takes two explicit arguments.

76) How an overloaded operator can be invoked using member functions?

In case of Unary operators, overloaded operator can be invoked as

op object_name or object_name op

In case of binary operators, it would be invoked as

Object . operator op(y)

where op is the overloaded operator and y is the argument.

77) How an overloaded operator can be invoked using Friend functions?

In case of unary operators, overloaded operator can be invoked as

Operator op (x);

In case of binary operators, overloaded operator can be invoked as

Operator op (x , y)

78) List out the operators that cannot be overloaded using Friend function.

_ Assignment operator =

_ Function call operator ( )

_ Subscripting operator [ ]

_ Class member access operator ®

79) What is meant by casting operator and write the general form of overloaded casting operator.

A casting operator is a function that satisfies the following conditions

_ It must be a class member.

_ It must not specify a return type.

_ It must not have any arguments.

The general form of overloaded casting operator is

operator type name ( )

{

…….. // function statements

}

It is also known as conversion function.

80) Explain basic to class type conversion with an example.

Conversion from basic data type to class type can be done in destination class. Using constructors does it. Constructor takes a single argument whose type is to be converted.

Eg: Converting int type to class type

```
class time

{

int hrs,mins;

public:

…….

Time ( int t) //constructor

{

hours= t/60 ; //t in minutes

mins =t % 60;

}

};
```

Constructor will be called automatically while creating objects so that this

conversion is done automatically.

81) Explain class to basic type conversion with an example.

Using Type Casting operator, conversion from class to basic type conversion can be done. It is done in the source class itself.

Eg: vector : : operator double( )

{

```
double sum=0;

for(int I=0;I<size;I++)

sum=sum+v[ i ] *u[ i ] ;

return sqrt ( sum ) ;

}
```

This function converts a vector to the corresponding scalar magnitude.

82) Explain one class to another class conversion with an example.

Conversion from one class type to another is the combination of class to basic and basic to class type conversion. Here constructor is used in destination class and casting operator function is used in source class.

Eg: objX = objY

objX is the object of class X and objY is an object of class Y. The class Y type data is converted into class X type data and the converted value is assigned to the obj X. Here class Y is the source class and class X is the destination class.

83) What is meant by inheritance?

Inheritance is the process by which objects of one class acquire the properties of another class. It supports the concept of hierarchical classification. It provides the idea of reusability. We can add additional features to an existing class without modifying it by deriving a new class from it.

84) What is meant by single inheritance?

If a single class is derived from a single base class is called single inheritance.

Eg:

Base class

Derived class

Here class A is the base class from which the class D is derived. Class D is the public derivation of class B hence it inherits all the public members of B. But D cannot access private members of B.

85) What is multiple inheritance?

If a class is derived from more than one base class, it is called multiple inheritance.

Eg: Base classes

Derived class

Here class C is derived from two base classes A & B.

86) What is hierarchical inheritance?

If a number of classes are derived from a single base class then it is called hierarchical inheritance.

Eg : Hierarchical classification of students in University

A

B

A

C

B

87) What is multilevel inheritance?

If a class is derived from a class, which in turn is derived from another class, is called multilevel inheritance. This process can be extended to any number of levels.

Eg:

Base class Grand father

Intermediate

Base class Father

Derived class Child

88) What is hybrid inheritance?

It is the combination of one or more types of inheritance.

Multilevel

inheritance

Multiple

inheritance

The class result will have both the multilevel and multiple inheritances.

Student

Arts Engineering M e d i c a l

CSE ECE Civil

A

B

C

Student

Test

Result

Sports

89) What is meant by Abstract base class?

A class that serves only as a base class from which derived classes are derived. No objects of an abstract base class are created. A base class that contains pure virtual function is an abstract base class.

90) Write short notes on virtual base class.

A base class that is qualified as virtual in the inheritance definition. In case of multiple inheritance, if the base class is not virtual the derived class will inherit more than one copy of members of the base class. For a virtual base class only one copy of members will be inherited regardless of number of inheritance paths between base class and derived class.

Eg: Processing of students' results. Assume that class sports derive the roll number from class student. Class test is derived from class Student. Class result is derived from class Test and sports.

As a virtual base class As a virtual base class

91)Define Polymorphism?

Polymorphism is the feature that allows one interface to be used for a general class of actions.(ie) "one interface multiple methods". This means that it is possible to design a generic interface to a group of related activites.This helps reduce complexity by allowing the same interface to be used to specify a general class of action.

92)Mention some of the Separators used in Java Programming?

( ) _ Contain a list of parameters in method definition & invocation.

{ } _ Contain the value of automatically initialized arrays.

[ ] _ Declare array types.

; _ Terminate statements.

. _ Separate package name from sub packages.

93)What is boolean data type?

Java has simple type called boolean for logical values. It can have only one of two possible values, true or false. This is the type returned by all relational operators like a<b.boolean is also required by the conditional expression that governs the control statements such as if for.

Student

Test

Result

Sports

Syntax: boolean variablename;

94)How dynamic initialization of variables is achieved in java?

Java allows variables to be initialized dynamically, using any expression valid at the time the variable is declared.

double a= 3.0,b=4.0

double c=Math.sqrt(a * a + b * b);

here "c" is initialized dynamically to the length of hypotenuse.

95)What is meant by Widening conversion?

When one type of data is assigned to another type of variable ,an automatic conversion will take place if the following conditions are met .

The two types are compatible.

The destination type is larger than the source type.

For example the int type is always large enough to hold to hold all byte values

96)What is meant by narrowing conversion?

We are explicitly making the value narrower so that it will fit into the target type. The conversion is not performed automatically. To create a conversion between two incompatible types, you must use a cast. A cast is simply an explicit type conversion.

Syntax: (target-type)value

target-type _ the desired type to convert the specified value to.

97)State Type Promotion Rules?

All byte & short values are promoted to int. if one operand is long ,the whole expression is promoted to long. If one operand is a float operand, the entire expression is promoted to float. If any one operand is double, the result is double.

byte, short_ long_ float _ double

98)How to create a one dimensional arrays?

A one dimensional array is a list of liked type variables. To create an array ,first create an array variable of desired data type.

Syntax: type var-name[];

99)Here var-name is set to null. To link with an actual, physical array of integers,

allocate using new.

bb: array-var=new type[size];

100)What is the use of ternary operator?

The ternary operator replaces if-then-else statements.

Syntax: expression1?expression2:expression3

Eg: ratio = denom = = 0 ? : num / denom;

If expression1 is true ,then expression2 is evaluated; otherwise expression3 is evaluated.

The result of ? operation is that of the expression evaluated.

101)Write down the syntax of switch statement?

Switch(expression){

case value1:

//statement sequence

break;

case value2:

//statement sequence

break;

.

.

.

case valueN:

//statement sequence

break;

default:

//default statement sequence

}

102)What are the iteration statements used in Java?

While: repeats a statement or block while its controlling expression is true.

Syntax: while(condition){

//body of loop

}

do-while: Executes its body atleast once

Syntax: do{

//body of loop

}while(condition);

for: consists of three portions initialization,conditon,termination.

Syntax: for(initialization,conditon,termination.){

//body

}

103)What is the difference between break & continue statements?

Break: We can force immediate termination of a loop, bypassing the conditional, the loop expression & any remaining code in the body of the loop. When a break statement is encountered in a loop, the loop is terminated & the program control resumes at the next statement following the loop.

Continue: useful to force early termination. it continue running the loop, but stop processing the remainder of the code in it's body for this particular iteration

104)What are the uses of break statements?

1.It terminates a statement sequence in switch statement.

2.It can be used to exit a loop.

3.it can be used as a civilized form of goto.

105)Define Constructors?

A constructor initializes an object immediately upon creation. It has the same names the class in which it resides & is syntactically similar to a method. Once defined the constructor is automatically called immediately after the object is created, before the new operator completes.

Syntax: class-var = new classname( );

106)Define parameterized Constructors?

To construct an object of various dimensions we can add parameters to the constructor.

Eg: Box mybox1= new Box(10,20,15);

Box mybox1= new Box(7,15);

107)What is the use of This keyword?

This is always a reference to the object on which the method was invoked. This can be used inside any method to refer to the current object.

```
Box(double w,double h,double d){

This. width = w;

This. height= h;

This. depth = d;

}
```

108)Define Garbage collection?

The technique used to handle the deal location automatically. When no references to an object exists, that object is assumed to be no longer needed,& the memory occupied by the object can be reclaimed. Garbage collection occurs sporadically during the execution of your program.

109)What is the use of finalize() method?

If an object is holding some non-java resource such as a file handle, or a window character font which might be freed before an object is destroyed. To handle such situation Java provides a mechanism called finalization. By using finalization we can define specific action that will occur when an objects just about to reclaimed by the Garbage collector.

Syntax: protected void finalize()

{

//finalization code

}

110)Define Method overloading?

In Java it is possible to define two or methods with the same class that share the same name, as long as the parameter declarations are different. When this is the case the methods are said to be overloaded &the process is referred to as method overloading.

void test(int a){

System.out.println("a: " +a);

}

void test(double a){

System.out.println("a: " +a);

}

void test(int a,int b){

System.out.println("a and b: " +a +" " +b);

}

111)What are the different ways of argument Passing?

Call-by-value: This method copies the value of an argument in to the formal parameter of the subroutine. Therefore changes made to the parameter of the subroutine have no effect on the argument used to call it.

Call-by-Reference: In this method ,a reference to an argument is passed to the parameter. Inside this subroutine ,this reference is used to access the actual argument specified in the call. This means that changes made to the parameter will affect the argument used to call the subroutine.

112)Mention some of the restrictions while using static keyword?

They can call other static methods.

They must only access the static data.

They cannot refer to **this** or **super** any way.

113)What is the use of final keyword?

It is used to prevent its contents from being modified. It is similar to const in C/C++.We must initialize a final variable when it is declared.

Eg: **final** int FILE_NEW=1;

114)How to call a Super class constructors?

A subclass can call a constructor method defined by its super class by use of the following form of **super.**

Super(parameter-list);

Parameter list_any parameters needed by the constructor in the superclass.super() must always be the first statement executed inside a subclass's constructor.

115)What are the uses of super keyword?

1.Using super we can call Super class constructor.

2.it acts like this, except that it always refers to the super class of the subclass in which it is used.

116)Define Method overriding?

When a method in a subclass has the same name & type signature as a method in the subclass is said to override the method in the superclass.when an overridden method is called from within a subclass it will always refer to the version of that method defined by the subclass.

```
class A{

…….

Void show(){

System.out.println(“Super class”)

}

class B extends A{

…….

Void show(){

System.out.println(“sub class”)

}

class override{

B sub = new B( );

Sub. show( );

}
```

117)Define Dynamic Method Dispatch?

Dynamic Method Dispatch is the mechanism by which a call to an overridden function is resolved at run time, rather than compile time. Java implements run time Polymorphism by means of Dynamic Method Dispatch.

Principle: A super class reference variable can refer to a sub class object. Java uses this to resolve calls to overridden methods at run time. when ah overridden method is called through a

super class reference, java determines which version of that method to execute based up on the type of the object being referred to at the time the call occurs. It is the type of the object being referred to that determines which version of an overridden method will be executed.

118)What are the uses of final keyword?

1. to create the equivalent of named constant.

2. To Prevent overriding

3. To prevent inheritance.

119)How to define a Package?

Include package statement as the first statement in a java sourcefile.The package statement defines a name space in which classes are stored.

Syntax: package pkg;

Pkg _ name of the package.

We can create a hierarchy of classes. For that separate each package name from the one above it by use of a period.

Syntax: package pkg1[pkg2[.pkg3]];

120)How to import packages?

Java includes the import statements to bring certain classes ,or entire packages in to visibility. import statements occur immediately following the package statements & before any class definitions.

Syntax: import pkg1[.pkg2].(class name| *);

Pkg1 _name of the Top level package.

pkg2 _name of the subordinate package inside the outer package separated by a

dot.

121)Write down the syntax for defining Interface?

An interface is defined similar to a class

Syntax: access interfacename{

Return type method-name1(parameter list);

Return type method-name2(parameter list);

type final –varname1 = value;

type final –varname1 = value;

//…

Return type method-nameN(parameter list);

type final –varname1 = value;

}

122)What are the steps to be followed while implementing interfaces?

To implement an interface, include the implements clause in a class definition,& then create the methods defined by the interface.

Syntax: access class classname[extends super class]

Implements interface[,interface…]]{

//class body

}

access _ either public or not used.

123)Write down the fundamentals of Exception Handling?

A java exception is an object that describes an exceptional condition that has occurred in a piece of code. When an exceptional condition arises an object representing that exception is created & thrown in the method that caused error.

Java Exception handling is managed through five keywords:try,catch,throw,

throws, and finally.

Syntax: try{

//block of code to monitor errors

}

catch(Expression type1 exob){

//exception handler for Exception type1

}

catch(Expression type1 exob){

//exception handler for Exception type1

}

//….

finally{

//block of code to be executed before try block ends.

}

124)Define Multithreaded Programming?

A Multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread ,And each thread defines a separate path of execution. Thus Multi threading is a specialized form of multitasking.

125)Mention some of the methods defined by Thread class?

Method Meaning

getName Obtain a thread's name

getPriority Obtain a thread's priority

isAlive Determine if a thread is still running

join wait for a thread to terminate

run Entry point for the thread

sleep Suspend a thread for a period of time.

126)What are the two ways for creating a Thread

By implementing the runnable interface.

Class Newthread implements Runnable

By Extending the Thread class

Class Newthread extends Thread

Part - B

1.What are the Features of Oop's & how are they implemented in C++?

• Objects.

• Classes.

• Data abstraction and Encapsulation.

• Inheritance.

• Polymorphism.

• Dynamic binding.

• Message passing.

2. Explain about inline function?

An inline function is a function that is expanded in line when it is invoked. That is compiler replaces the function call with the corresponding function code.The inline functions are defined as

Inline function-header

{

function body

}

The situations where inline expansion may not work are

• for functions returning values, if loop, a **switch,** or a **goto** exists

• for functions not returning values ,if a return statement exists

• if function contain **static** variables

• if **inline** functions are recursive

3. Explain Function Overloading?

Function overloading means we can use the same function name to create functions that perform a variety of different tasks.

Eg: An overloaded add ( ) function handles different data types as shown below.

// Declarations

iv. int add(int a, int b); //add function with 2 arguments of same type

v. int add(int a, int b, int c); //add function with 3 arguments of same type

vi. double add(int p, double q); //add function with 2 arguments of

different type

//Function calls

i. add (3 , 4);

ii. add (3, 4, 5);

iii. add (3 , 10.0);

4. Explain about Operator Overloading?

C++ has the ability to provide the operators with a special meaning for a data type. This mechanism of giving such special meanings to an operator is known as Operator overloading. It provides a flexible option for the creation of new definitions for C++ operators.

The operators that cannot be overloaded are.

• Class member access operator (. , .*)

• Scope resolution operator (::)

• Size operator (size of)

• Conditional operator (?:)

The purpose of using operator function is to define an additional task to an operator, we must specify what it means in relation to the class to which the operator is applied. This is done by Operator function , which describes the task. Operator functions are either member functions or friend functions.

The general form is

return type classname :: operator (op-arglist)

{

function body

}

where *return type* is the type of value returned by specified operation.

Op-operator being overloaded. The *op* is preceded by a keyword operator. operator op is

the function name. The rules for Operator overloading are

• Only the existing operators can be overloaded.

• The overloaded operator must have at least one operand that is of user

defined data type.

• The basic meaning of the operator should not be changed.

• Overloaded operators follow the syntax rules of the original operators.

They cannot be overridden.

5. Explain overloading Unary & Binary operator?

When unary operators are overloaded using member functions it takes no explicit arguments and return no explicit values.

When binary operators are overloaded using member functions, it takes one explicit argument. Also the left hand side operand must be an object of the relevant class.

When unary operators are overloaded using friend function, it takes one reference argument (object of the relevant class)

When binary operators are overloaded using friend function, it takes two explicit arguments. The operator can be invoked using member functions as follows

In case of Unary operators, overloaded operator can be invoked as op object_name or object_name op

In case of binary operators, it would be invoked as

Object . operator op(y)

where op is the overloaded operator and y is the argument.

The overloaded operator can be invoked using Friend function as

In case of unary operators, overloaded operator can be invoked as

Operator op (x);

In case of binary operators, overloaded operator can be invoked as

Operator op (x, y)

The operators that cannot be overloaded using Friend function.

• Assignment operator =

• Function call operator ( )

• Subscripting operator [ ]

• Class member access operator ®

6. Explain about Type conversions?

The three types of data conversion are

i. Conversion from basic type to class type.

ii. Conversion from class type to basic type.

iii. Conversion from one class type to another class type.

A casting operator is a function that satisfies the following conditions

• It must be a class member.

• It must not specify a return type.

• It must not have any arguments.

The general form of overloaded casting operator is

operator type name ( )

{

…….. // function statements

}

It is also known as conversion function.

i. Basic to class type conversion

Conversion from basic data type to class type can be done in destination class.Using constructors does it. Constructor takes a single argument whose type is to be converted.

Eg: Converting int type to class type

```
class time

{

int hrs,mins;

public:

…….

Time ( int t) //constructor

{

hours= t/60 ; //t in minutes

mins =t % 60;

}

};
```

Constructor will be called automatically while creating objects so that this conversion is done automatically.

ii. Basic type conversion with an example.

Using Type Casting operator, conversion from class to basic type conversion can be done. It is done in the source class itself.

Eg: vector : : operator double( )

```
{

double sum=0;

for(int I=0;I<size;I++)

sum=sum+v[ i ] *u[ i ] ;

return sqrt ( sum ) ;
```

}

This function converts a vector to the corresponding scalar magnitude.

iii. One class to another class conversion with an example.

Conversion from one class type to another is the combination of class to basic and basic to class type conversion. Here constructor is used in destination class and casting operator function is used in source class.

Eg: objX = objY

objX is the object of class X and objY is an object of class Y. The class Y type data is converted into class X type data and the converted value is assigned to the obj X. Here class Y is the source class and class X is the destination class.

7. Explain inheritance?

Inheritance is the process by which objects of one class acquire the properties of another class. It supports the concept of hierarchical classification. It provides the idea of reusability. We can add additional features to an existing class without modifying it by deriving a new class from it.

i. single inheritance

If a single class is derived from a single base class is called single inheritance.

Eg:

Base class

Derived class

Here class A is the base class from which the class D is derived. Class D is the public derivation of class B hence it inherits all the public members of B. But D cannot access private members of B.

ii. multiple inheritance

If a class is derived from more than one base class, it is called multiple inheritance.

Eg: Base classes

Derived class

Here class C is derived from two base classes A & B.

iii. Hierarchical inheritance

If a number of classes are derived from a single base class then it is called hierarchical inheritance.

Eg : Hierarchical classification of students in University

A

B

A

C

B

Student

Arts Engineering M e d i c a l

CSE ECE Civil

iv.Multilevel inheritance

If a class is derived from a class, which in turn is derived from another class, is called multilevel inheritance. This process can be extended to any number of levels.

Eg:

Base class Grand father

Intermediate

Base class Father

Derived class Child

v. Hybrid inheritance

It is the combination of one or more types of inheritance. The class result will have both the multilevel and multiple inheritances.

Multilevel

inheritance

Multiple

inheritance

8. Define constructor

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

Eg:

Class **integer**

{

……

public:

integer( );//constructor

……

}

A

B

C

Student

Test

Result

Sports

The different types of constructor are

i. default constructor

The constructor with no arguments is called default constructor

Eg:

```
Class integer
{
int m,n;
Public:
Integer( );
…….
};
integer::integer( )//default constructor
{
m=0;n=0;
}
```

the statement

integer a;

invokes the default constructor

ii. parameterized constructor

constructor with arguments is called parameterized constructor

Eg;

```
Class integer
{ int m,n;
public:
```

integer(int x,int y)

{ m=x;n=y;

}

To invoke parameterized constructor we must pass the initial values as arguments to the constructor function when an object is declared. This is done in two ways

1.By calling the constructor explicitly

eg:

integer int1=integer(10,10);

2.By calling the constructor implicitly

eg:

Integer int1(10,10);

iii. Default argument constructor

The constructor with default arguments are called default argument constructor

Eg:

Complex(float real,float imag=0);

The default value of the argument imag is 0

The statement

complex a(6.0)

assign real=6.0 and imag=0

the statement

complex a(2.3,9.0)

assign real=2.3 and imag=9.0

iv. Copy constructor

A copy constructor is used to declare and initialize an object from another object. It takes a reference to an object of the same class as an argument

Eg: integer i2(i1);

would define the object i2 at the same time initialize it to the values of i1.

Another form of this statement is

Eg: integer i2=i1;

The process of initializing through a copy constructor is known as **copy initialization**.

v.Dynamic constructor

Allocation of memory to objects at time of their construction is known as dynamic constructor. The memory is allocated with the help of the **NEW operator**

Eg:

```
Class string

{

char *name;

int length;

public:

string( )

{

length=0;

name=new char[length +1];

}

void main( )

{

string name1(“Louis”),name3(Lagrange);
```

}

use delete to free that memory.

9. Explain about Multiple constructors (constructor overloading)?

The class that has different types of constructor is called multiple constructors

Eg:

```
#include<iostream.h>
#include<conio.h>
class integer
{
int m,n;
public:
integer( ) //default constructor
{
m=0;n=0;
}
integer(int a,int b) //parameterized constructor
{
m=a; n=b;
}
integer(&i) //copy constructor
{
m=i.m;
```

```
n=i.n;

}

void main()

{

integer i1; //invokes default constructor

integer i2(45,67);//invokes parameterized constructor

integer i3(i2); //invokes copy constructor

}
```

The special characteristics of constructor are

• **T**hey should be declared in the public section

• They are invoked automatically when the objects are created

• They do not have return types, not even void and therefore, and they cannot return values

• They cannot be inherited, though a derived class can call the base class

• They can have default arguments

• Constructors cannot be virtual function

10. Explain virtual functions

A function qualified by the 'virtual' keyword is called virtual function. When a virtual function is called through a pointer, class of the object pointed to determine which function definition will be used.

The rules for virtual functions are

_ Virtual functions must be member of some class.

_ They cannot be static members and they are accessed by using object pointers

_ Virtual function in a base class must be defined.

_ Prototypes of base class version of a virtual function and all the derived class versions must be identical.

_ If a virtual function is defined in the base class, it need not be redefined in the derived class.

Pure virtual functions

A pure virtual function is a function declared in a base class that has no definition relative to the base class. In such cases, the compiler requires each derived class to either define the function or redeclare it as a pure virtual function. A class containing pure virtual functions cannot be used to declare any object of its own. It is also known as “donothing” function.

The “do-nothing” function is defined as follows:

virtual void display ( ) =0;

11. Explain the features of Java?

• Compiled and Interpreted

• Platform-Independent and portable

• Object oriented

• Robust and Secure

• Distributed

• Familiar, simple and small

• Multithreaded and Interactive

• High Performance

• Dynamic and Extensible

12. Describe the structure of Java program?

• Documentation section

• Package statement

• Import statements

• Interface statements

• Class definitions

• Main method class

13. Explain about Inheritance in Java?

• Defining a subclass

• Subclass constructor

• Single Inheritance

• Multilevel Inheritance

• Hierarchical Inheritance

Eg:

```
Class A
{
---------
---------
}
class B extends A
{
---------
---------
}
```

class C extends B

{

}

14. Explain about Interfaces in Java?

Java does not support multiple inheritances. It is achieved using interfaces in Java.

• Defining Interfaces

• Extending Interfaces

• Implementing Interfaces

• Accessing Interface variables

Eg :

Interface Area

{

final static flaot pi=3.14f;

float compute(float x,float y);

}

15. Explain about Packages?

Packages are java's way of grouping variety of classes or interfaces together.

Packages act as containers for classes

• Java API Packages

• Using System packages

• Creating Packages

• Accessing a package

• Using a package

Eg:

```
Package p1;

Public class A

{

public void display()

{

System.out,println(“Class A”);

}

}
```

16. Explain about Threads in Java?

A thread is a program that has a single flow of control

• Creating threads

• Extending the thread class

Eg:

```
Class Mythread extends Thread

{

------

------
```

}

• Using Runnable Interface

Eg:

Class MyThread implements Runnable

{

}

• Stopping and Blocking a thread

17. Explain about Strings in Java?

• Strings can be created using

_ String class

String s= new String(“Hello”);

_ StringBuffer class

StringBuffer s= new StringBuffer(“Hello”);

• String arrays

• String methods

18. Explain about Thread lifecycle?

• Newborn state

• Runnable state

• Running state

• Blocked state

• Dead state

19. Explain about Exception handling in Java?

• Exceptions

• Syntax of Exceptions handling state

• Multiple catch statements

• Using finally statement

Eg:

```
Class A
{
public static void main (String args[])
{
int a=10;
int b=5;
int c=5;
int x,y;
try
{
x=a/(b-c);
}
catch(ArithmeticException e)
{
```

```
System.out.println(“Division by zero”);
}
y=a/b+c;
System.out.println(“y=”+y);
}
}
```

20. Explain about Applet Lifecycle?

• Initialization state

• Running state

• Idle state or stopped state

• Dead state

• Display state

21. How Applets are prepared and executed?

• Writing Applets

Eg:

```
Import java.awt.*;
Import java.applet.*;
Public class A extends Applet
{
public void paint( Graphics g)
{
```

g.drawString(“Hello”,10,100);

}

}

• Building Applet code

• Designing a web page

<html>

<title>Applet</title>

<body>

<applet code= A.class width=400 height=200 </applet>

</body>

</html>

• Running the Applet

• Using appletviewer

• Using web browser

22. Explain about Visibility Controls?

• Public access

• Friendly access

• Protected access

• Private access

• Private protected access

23. Explain about Methods overriding and methods overloading?

Methods that have same name but different parameter lists and different definitions is called Method overloading.

Eg:

```
class Room
{
int width;
int length;
Room(int x,int y) // Constructor
{
length= x;
width = y;
}
Room(int x)
{
length=breadth=x;
}
}
```

When a method is called the method defined in the subclass is invoked and executed instead of the one in the superclass. This is called overriding.

Eg:

```
class superclass
{
void methodsuper()
```

```
{
System.out.println(\"superclass method\");
}
}
class subclass extends superclass
{
void methodsuper()
{
System.out.println(\"overriding method of superclass\");
}
}
class override
{
public static void main(String args[])
{
subclass sb=new subclass();
sb.methodsuper();
}
}
```

24 Explain about operators in Java?

• Arithmetic operators

• Integer arithmetic

• Real arithmetic

• Mixed mode arithmetic

• Relational operators

• Logical operators

• Assignment operators

• Increment and Decrement operators

• Conditional Operator

• Bitwise operator

• Special operators

• Instanceof operator

• Dot operator

25 Explain about Constructors in Java?

Constructors enable an object to initialize itself when it is created.

Eg: class Rect

{

int width;

int length;

Rect(int x,int y) // Constructor

{

length= x;

width = y;

}

```
int rectArea()
{
return(length *width);}
}
```

26. Explain the various Java tokens?

• Java character set

• Keywords

• Identifiers

• Literals

• Operators

• Separators

27. How will you implement a Java program?

• Creating the Java program

```
class sample
{
public static void main(String args[])
{
System.out.println(“Hello”);
}
}
```

Save the program as sample.java

• Compiling the program

The program is compiled using the statement and it will create a

class file named sample.class.

Javac sample.java

• Running the program

Java sample

This statement will execute the program and display the output.

28. What are the features of simple Java program?

• Class declaration

• Opening brace

• Main Line

• Output line

Eg:

class sample // Class declaration

{ // Opening brace

public static void main(String args[]) // Main Line

{

System.out.println(“Hello”); // Output line

}

}_