OOPS & C++

- Programming in C++ uses the concept of object oriented programming. The style of object oriented programming is something new compared to other programming style.
- All programming methodologies are classified into 3 styles as

1. Low Level Programming
   - machine level language
   - assembly level language
2. High Level Programming
   - Procedural/Structured programming
     eg. C, Pascal, FORTRAN ,COBOL
   - Functional Programming
     Eg. LISP
   - Logic Programming
     Eg. PROLOG
   - Object-oriented Programming
     Eg. Java, smalltalk, C++
3. Very High Level Programming
   - Command Language
     Eg. Shell programming in Unix
   - Query Language
     Eg. SQL in RDBMS

Advantages of procedural programming

- Easy to learn and teach
- Mainly used for general purpose
- Easy to design for solution, easy to verify the functionality
- Testing and maintenance is possible

Disadvantages of procedural programming

- Maintenance and modification is possible but very difficult for large applications.
- Data cannot be hidden

Features of Object-oriented Programming

- Improvement over the structured programming paradigm
- Emphasis on data rather than algorithm
- Data abstraction is introduced in addition to procedural abstraction
- Data and associated operations are unified into a single unit, thus the objects are grouped with common attributes, operations and semantics.
- Programs are designed around the data being operated, rather than operations themselves.
- Eg. C++,Smalltalk, Eiffel, Java etc.

- Basic concepts of OOP
  - Objects

- Classes
- Data abstraction
- Data Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing
- Extensibility
- Persistence
- Delegation
- Genericity

- Objects : Objects are basic runtime entities in object oriented system. An object is an instance of a class. They represent person, place, bank account etc.
- Classes : A class is a template or model by which any number of objects can be created. A class is a collection of objects of similar type. It defines the characteristics and operations.
- Data Abstraction : The technique of creating new data types that are well suited to an application to be programmed is known as data abstraction. It is a process of hiding unnecessary details and taking up only wanted details.
- Data Encapsulation : It is a mechanism that associates the code and the data it manipulates into a single unit and keeps them safe from external interference and misuse. In C++, this is supported by a construct called class.
- Inheritance : It allows the extension and reuse of existing code without having to rewrite the code from scratch. Inheritance involves the creation of new classes(derived classes) from the existing ones(base classes) thus enabling the creation of a hierarchy of classes that simulate the class and subclass concept of the real world.
- Polymorphism : It allows a single name/operator to be associated with different operations depending on the type of data passed to it. In C++, it is achieved by operator overloading and function overloading.
- Dynamic Binding : The process of binding the code at runtime to the function calls is called dynamic binding.
- Message Passing : It is the process of invoking an operation on an object. In response to a message, the corresponding method is executed in the object.
- Extensibility : It is the feature which allows the extension of the functionality of the existing software components.
- Persistence : The phenomenon where the object outlives the program execution time and exists even after the execution of the program is known as persistence.
- Delegation : It is alternative to inheritance. Delegation is a way of making object composition as powerful as inheritance.
- Genericity : It is the property used for defining functions and classes of the same functionality and which differ only in the data type.
- Depending on the object features supported , the languages are classified into two categories

I. Object based programming languages

II. Object Oriented programming languages

- Object based programming languages support encapsulation and object identity without supporting important features of OOP languages such as polymorphism ,inheritance, message passing. Ada is a typical example of object based programming language.
- Object oriented programming languages incorporate all the features of object-based programming languages along with inheritance and polymorphism.
- C++ is an object oriented programming language developed by Bjarne Stroustrup at AT & T Bell laboratories.
- Stroustrup combined the features of C and simula 67.
- C++ is a superset of C and is also called "C with classes".
- C++ supports a rich set of functions for performing input and output operations.
- C++ handles I/O operations by using streams.
- Streams in C++ are classified into
  - Output Streams
  - Input Streams
- Output Streams : The output streams allow to perform write operations on output devices such as screen, disk etc.
- Output on the standard stream is performed using the cout object.
- The word cout is followed by the symbol << which is called insertion or put-to operator.
- More than one item can be displayed using a single cout output stream object. Such operations in C++ are called cascaded output operations.
- The cout object will display all the items from left to right.
- Input Streams : The input streams allow to perform read operation with input devices such as keyboard, disk etc.
- Input from the standard stream is performed using the cin object.
- Cin is followed by the symbol >> which is called the extraction operator.
- Input of more than one item which can be of any type can be performed by using a single cin object. Such operations in C++ are cascaded input operations.
- Streams do not require explicit data type specification and explicit address operator.
- << and >> are also used as left shift and right shift operators.
- C++ supports single line comments // along with the ordinary comments in C(/* */).
- C++ supports a mechanism to access a global variable from a function in which a local variable is defined with the same name as a global variable. It is achieved by using scope resolution operator (::).
- Unlike in C, variables can be defined in any part of the program in C++.
- Variable Aliases or Reference variables
- Reference variables enjoy the simplicity of value variables and the power of pointer variables. They are also called aliases.
- They are created by using the symbol & before the reference variable name.
- Syntax : datatype & referencevariable = variablename;
- The reference variable must be initialized to some variable only at the point of its declaration.

- Constants cannot be pointed by reference variables.
- C++ is a strict type checking language. Function prototyping is compulsory.
- Inline Functions :
- These are also called as open sub-routines similar to macros.
- Inline functions are used when the function is very small.
- The keyword inline is used before the function to define it as inline.
- Any function defined within a class is by default inline.
- The function call is replaced by the function code by the compiler.
- So inline functions generally increase the compilation time and decrease the execution time.
- The function defined as inline need not be definitely substituted , it is compiler dependent(the compiler takes the decision).
- In general, inline functions should not be used . Only the functions which contain one simple statement should be defined as inline.
- Default Arguments :
- In a function call, when one or more arguments are omitted, the function may be defined to take default values for omitted arguments by providing the default values in the function prototype.
- The default arguments can be given from right to left.
- Default arguments reduces the burden of passing arguments explicitly at the point of function call.
- The feature of default arguments can be utilized to enhance the functionality of the program, without the need for modifying the old code referencing to functions.
- All the arguments in a multiple argument function need not have default values.
- Type Conversion :
- The feature of the compiler that performs data conversion without the user intervention, is known as implicit type conversion.
- Type casting in C++ can be done explicitly by using any of the given syntax
  (data type) variable or data type (variable)
- Runtime Memory management:
- If the amount of memory required is unknown at the compile time then the memory allocation can be performed during execution. Such allocation is called dynamic memory allocation.
- C++ provides two special operators to perform memory management dynamically. They are new and delete operators.
- new operator : It is used for allocating memory dynamically similar to malloc() function.
- It throws an exception if memory allocation fails.
- Syntax : datatype * new datatype[size]
- new operator can be overloaded.
- The memory allocated by new operator cannot be deallocated by free() function.
- delete operator : It is used for deallocating the memory allocated by new operator.
- **Syntax :** **delete** pointervariable;

- The pointervariable will not be deleted but the memory pointed by that variable will be released.
- The **delete** operator cannot be used for deallocating memory allocated by **malloc()** function.
- **delete** operator can also be overloaded.
- To delete an array of integers or float or double the syntax is

  delete [] arrayname;
- **Classes and Objects :**
- Object oriented programming constructs are modeled out of the data types called **classes.**
- Defining variables of a class datatype is known as a class instantiation and such variables are called objects.
- A class encloses both the data and functions that operate on the data, into a single unit.
- The variables and functions enclosed in a class are called *data members* and *member functions.*
- Classes are similar to structures. The difference between structures and classes of C++ is that in a structure by default the members are public whereas the members within the class are private by default.
- Class that do not contain data members and member functions are called empty classes. Empty classes are also called stubs.
- Members within the class can be *private or public or protected.* These are called access specifiers.
- The private members have strict access control and only the member functions of the same class can be accessed.
- The private members of a class are not accessible outside the class except through friend functions. So friend functions hinder the concept of data hiding.
- The access control of protected members is similar to that of private members and has more significance in inheritance.
- The members defined as protected can be accessed by its members and members of its subclasses.
- The members of the class declared as public can be accessed without restriction even from outside the class.
- The public, private and protected sections are defined within the class by using the keywords **public, private and protected.**
- The public , private and protected sections within a class can be in any order.
- Data is hidden inside a class and it is achieved by declaring the data as *private.*
- The syntax for defining a class is

  class class name {

  };
- The data members must be declared within the body of the class, whereas the member functions can be defined either inside the class or outside the class.
- If it is to be defined outside the class , the class name and scope resolution operator(::) are used along with it to mention that it belongs to a particular class.
- Functions defined inside the class are by default inline whereas the functions defined outside the class are outline.

- A class can have multiple member functions with the same name but they should differ in the number or type of arguments. It is called function overloading.
- Two classes within a same program cannot have same names.
- The resources are not allocated when a class is defined but they are allocated when the class is instantiated i.e. when the object is created.
- Classes can have pointers.
- Even it can have pointer which points to object of same classes. Such classes are called self-referential classes.
- Self-referential classes are used for implementing complicated data structures like trees, graphs and linked lists.
- The data members and member functions of an object can be accessed by using (.)dot operator. objectname. Membername
- Objects can also be passed as arguments to functions and as well they can be returned by a function.
- **Friend Functions:**
- If a function which is not a member of a class and wants to access the private members of a class then they should be friend functions.
- A function can be friend for multiple classes.
- The scope of a friend function is not limited to the class in which it has been declared as a friend.

  A friend function cannot be called using the object of that class; it is not the scope of the class. It can be invoked like a normal function without the use of any object.
- Unlike class member functions , it cannot access the class members directly.
- It can be declared in the private part or public part.
- Friendship is not mutual by default. If B is declared as a friend of A, this does not mean A is a friend of B.
- Friend functions are used as bridge for two or more classes.
- Friend functions are also used to increase the versatility of overloaded operators.
- Friendship is not transitive.
- **Friend classes :**
- All the member functions of one class can be friend of another class.
- A specific member function of one class can be friend function to another class.
- **Static data members and functions:**
- The static data members must be initialized during their definition outside all the member functions in the same way as global variables are initialized.
- Irrespective of whether the data member is private, public or protected , it must be defined using the scope resolution operator.
- Static variables act like a bridge between objects of the same class.
- Static functions can access only static members defined in the same class, non-static data are unavailable to these functions.
- Only one copy of static data member exists for all the instances of the class.
- Static data members can be accessed even before any object is created by using the class name.

- Storage space for data members which are declared as static is allocated only once during the class declaration.
- **Constructors:**
- A constructor enables an object to initialize itself during creation. This operation is called object initialization.
- A constructor is a special member function whose main operation is to allocate required resources such as memory and initialize the objects of its class.
- A constructor has the same name as that of the class to which it belongs.
- A constructor is executed automatically whenever the class is instantiated.
- A constructor does not have any return type.
- It is normally used to initialize the data members of a class.
- Constructors can be invoked with any number of functions .
- Constructors can be overloaded as ordinary functions.
- Constructor without any arguments is called a default constructor.
- Constructors can be virtual
- Constructors can also be constructed by using default arguments.
- **Destructor:**
- The destructor destroys the object when it is no longer required, by releasing all the resources allocated to it. This operation is called clean up.
- It has the same name of the class and has a ~ symbol before it.
- A destructor is automatically invoked whenever the object is to be destroyed.
- A destructor does not return any value.
- A destructor cannot take arguments. So destructor overloading is not possible.
- Destructors can be virtual.
- There should be only one destructor for a single class.
- The scope of dynamically created data members is not within the class so the destructor should be explicitly called to destroy dynamically created memory.
- This reduces memory leaking.
- C++ supports the creation of unnamed objects or nameless objects.
- Nameless objects can be used within a single statement.
- A nameless object is created and destroyed at the same point.
- The objects which are defined as constant are called read-only objects.
- A class declared inside the declaration of another class is called nested class.
- Nesting of classes enables in building very powerful data structures.
- A object which is definitely initialized during the process of its construction is called a live object.
- A class whose live object is to be created must have at least one constructor.
- An array of objects can be created.
- **Copy Constructor:**
- A constructor having a reference to an instance of its own class as an argument is known as copy constructor.
- All the members of the source object are copied into destination object by the copy constructor.

- A copy constructor can have only one argument.
- A copy constructor can take an argument by pass by reference.
- The argument for copy constructor cannot be passed by value.
- If an argument is passed by value the constructor is called again and again until it is out of memory.

- **this pointer:**
- The keyword **this** is a pointer variable , which always contains the address of the object in question.
- Each member function of a class is born with a pointer called **this** , which points to the object itself.
- The **this** pointer is used to access even the data members inside.
- The **this** pointer cannot be used within a function which is not the member of any class.
- **this** pointer is used in member functions returning pointers to their respective objects.
- It is also used to remove ambiguities between class data members and arguments of the function if they have the same name.
- **Operator Overloading:**
- Operator overloading is used for
  - i. Extending the capability of operators to operate on user defined data.
  - ii. Data conversion.
- Operator overloading is a feature which supports polymorphism.
- Many operators can be overloaded in C++.
- At least one operand of the operator which is overloaded should be an object of a class.
- The precedence relation of over loadable operators and their expression syntax remains the same.
- Only existing operators can be given new functionality but new operators cannot be defined.
- An operator is overloaded in a class by defining **operator** function by using **operator** keyword.
- An unary operator overloading function requires no arguments.
- A binary operator overloading function requires one argument.
- Ternary operator cannot be overloaded.
- The operator function can be defined within the class or outside the class like any other member function.
- In the overloading of binary operators, the left hand operand is used to invoke the operator function and the right hand operand is passed as an argument to the operator function.
- The operators **new** and **delete** can be overloaded.
- **Data Conversion:**
- The compiler can convert one of the basic type to another basic type automatically. This is known as *implicit type conversion.*
- Conversion can be explicitly specified by type casting. It is known *explicit type conversion.*
- Conversion from user defined data type to basic type is done with the help of a conversion function.

- Conversion from basic data type to user-defined type is done with the help of one-argument constructor.
- Conversion from one user defined type to another user defined type can be done by using an one argument constructor or conversion function.
- A class can have any number of conversion functions.
- Operators can be overloaded as friend operator functions.
- The operators which cannot be overloaded as friend operator functions but can be overloaded as operator member functions are =, 0,[],->.
- The operators which cannot be overloaded at all are

.	(dot operator)
::	(scope resolution operator)
?:	(conditional operator)
sizeof	(size of operator)
->*	(pointer to member)

- **Inheritance:**
- New classes can be built from existing classes. The technique of building new classes from the existing classes is called inheritance.
- The new classes are called derived classes or children or subclasses or descendent where as the existing classes are called base classes or parent or super classes or ancestors.
- The derivation of *derived class* from the *base class* is indicated by :(colon)
- Visibility mode specifies whether the features of the base class are publicly or privately inherited.
- In public derivation, the public members of the super class will become public in derived class and the protected members of the super class will become protected.
- In private derivation, the public members of the super class will become private in derived class and the protected members of the super class will become private.
- A derived class inherits data members and member functions , but not the constructor or destructor from its base class.
- The different forms of inheritance are
  i. single inheritance
  ii. multiple inheritance
  iii. hierarchical inheritance
  iv. multilevel inheritance
  v. hybrid inheritance
  vi. multipath inheritance
- **Single inheritance :** Derivation of a class from only one base class is called single inheritance.
- **Multiple inheritance:** Derivation of a class from several base classes is called multiple inheritance. It causes ambiguities and problems.
- **Hierarchical inheritance :** Derivation of several classes from a single base class is called hierarchical inheritance.
- **Multi-level inheritance :** Derivation of a class from another derived class is called multi-level inheritance.

- **Hybrid inheritance :** Derivation of a class involving more than one form of inheritance is called hybrid inheritance.
- **Multipath inheritance :** Derivation of a class from another derived classes , which are derived from the same base class is called multipath inheritance.
- If the base class has constructors with arguments then it is mandatory for the derived class to have a constructor.
- When an object of a derived class is created , the constructor of the base class is executed first and later the constructor of the derived class.
- In the derived class, first the constructors of virtual base classes are invoked and then non-virtual base classes.
- If the same member function or data exists in both the base class and the derived class the member of the derived class is executed.
- The default visibility mode of inheritance is private.
- Inheritance allows the construction of reusable software components.
- **Virtual Base classes :** The concept of virtual base classes is used to remove the duplication due to multipath inheritance.
- **Virtual Functions :** The concept of virtual functions is used to implement run-time polymorphism.
- Resolving a function call at compile time is called compile-time or static or early binding.
- Resolving a function call at runtime is called run-time or dynamic or late binding.
- Virtual functions postpone the decision of selecting the suitable member functions until runtime.
- A pointer to a base class can also point to any one of its derived classes.
- Virtual functions should be defined in the public part to realize its full potential.
- Virtual functions should be accessed through pointers of the base class.
- Virtual functions with null body or no definition are called pure virtual functions.
- These functions do nothing and are also called dummy functions.
- Pure virtual function is declared as a virtual function with its declaration followed by =0.
- A class with at least one pure virtual function is called an abstract class. It is not a complete class as the definition is not complete. So it cannot have instances of its own(i.e. objects cannot be created)
- A class without any pure virtual function is called concrete class.
- Destructors of a base class should be declared as virtual.
- Virtual functions cannot be static members.
- A virtual function can be a friend to another class.
- A class with a pure virtual function cannot be instantiated.
- **Templates:**
- Templates support generic programming.
- It allows to develop reusable software components such as functions, classes supporting different data types.
- **Function templates:**

- A function template or generic function is a function which used as a general function for various functions which have the same functionality but differ only in data types.
- Templates are defined by using the keyword **template.**
- A call to a template function is similar to that of a normal function and the parameters can be of any data-type .
- When the compiler encounters a call to such functions, it identifies the data type of the parameters and creates a function internally and makes a call to it.
- Function templates can be overloaded.
- A template function should definitely have an argument which is of generic type.
- **Class templates:**
- Classes which are declared to operate on different data types are called class templates.
- Any call to the template functions and classes, needs to be associated with a data type or a class.
- The compiler then instantiates a copy of the template function or template class for the data type specified.
- Class templates can be inherited.
- A derived class of a template based base class is not necessarily template derived class.
- Delegation can be implemented by template functions.
- **Stream computation with Console:**
- C++ uses the concept of streams and stream classes to perform I/O operations with console and disk files.
- Streams are classified into input streams and output streams.
- A stream is a series of bytes, which act either as a source from which input data can be extracted or as a destination to which the output can be sent.
- The C++ language offers a mechanism which permits the creation of an extensible and consistent input-output system in the form of streams library.
- The predefined streams in C++ are

Cin	standard input usually keyboard
Cout	standard output usually screen
Cerr	standard error output
Clog	fully buffered version of cerr

- The main advantage of using iostream.h functions over the stdio.h functions is data independence.
- Mixed usage of stdio and stream class functions are not advisable.
- Cin and cout are objects of certain classes defined in iostream.h.
- The C++ input-output system supports a hierarchy of classes that are used to manipulate both the console and disk files, called stream classes.
- **ios class** provides operations common to both input and output.
- The classes derived from ios class are
  - istream
  - ostream
  - iostream
- **istream** is a derived class of ios and inherits the properties of ios. It defines input functions such as get(),getline() and read(). The stream extraction operator >> is overloaded in it.

- **ostream** is derived class of ios and inherits the properties of ios. It defines output functions such as put() and write(). The stream insertion operator << is overloaded in it.
- **iostream** is derived from multiple base classes , **istream** and **ostream** , which in turn are inherited from the class ios. It provides facility for handling input and output streams.
- The function get() is a member function of the input stream class **istream** and is used to read a single character from the input device.
- The function put() is a member function of the output stream class **ostream** and is used to write a single character to the output device.
- The function getline() reads a whole line of text that ends with new line or until the maximum limit is reached.
- Formatted console operations can be performed by using
  1. ios stream class member functions and flags
  2. standard manipulators
  3. user defined manipulators
- **ios stream class member functions and flags:**
- The following the important ios class functions

Function	Task performed
i. width()	specifies the required number of fields to be used while displaying the output value.
ii. precision()	specifies the number of digits to be displayed after the decimal point.
iii. fill()	specifies a character to be used to fill the unused area of a field. By default, fills blank space character.
iv. setf()	sets format flag that control the form of output display
v. unsetf()	clears the specified flag.

- The following the flags of ios stream class

Flags value	Bit field	Effect produced
ios:: left	ios::adjustfield	left justified output
ios::right	ios::adjustfield	right justified output
ios::internal	ios::adjustfield	adding occurs between sign and no.
ios::dec	ios::basefield	decimal conversion
ios::oct	ios::basefield	octal conversion
ios::hex	ios::basefield	hexadecimal conversion
ios::scientific	ios::floatfield	using exponential floating notation
ios::fixed	ios::floatfield	using ordinary floating notation

- C++ has manipulators which produce output and consume input to extend stream I/O formatting. Such manipulators can be especially useful for simple parsing of stream inputs.
- Manipulators are categorized into the following two types

i. non-parameterized manipulators
ii. parameterized manipulators

- The manipulators are defined in iomanip.h.
- The following are some of the manipulators

Manipulator	Action performed
dec	sets the conversion base to 10
hex	sets the conversion base to 16
oct	sets the conversion base to 8
ws	extracts white space characters from input stream
endl	outputs a newline and flusthes stream
flush	flushes the stream
setw(width)	sets the field width
setprecision(prec)	sets floating point precision
setfill(fchar)	sets the fill character
setbase(base)	sets the conversion base
setiosflags(long)	sets the format flag
resetiosflags(long)	resets the format flag

- The user can also define his own manipulators.
- The user defined manipulator will be declared with an argument.
- In C++, streams are used for file computations.
- The three stream classes used for file handling are
  - ifstream for handling input files
  - ofstream for handling output files
  - fstream for handling files on which input and output can be performed
- **ifstream** class supports input operations. It opens the file in input mode by default. It inherits get() , getline(),read() functions from istream.
- **ofstream** class supports output operations. It opens the file in output mode by default. It inherits put(),seekp(),tellp() and write() functions from ostream.
- **fstream** supports simultaneous input and output operations. It opens the file for both input and output.
- In C++, a file can be opened using the following
  - Constructor function of the class
  - The member function open() of the class
- A file can be opened in the following modes

Mode	Effect of the mode
ios::in	open for reading
ios::out	open for writing
ios::ate	go to the end of the file at opening time
ios::app	open in append mode
ios::trunc	trucate the file if it already exists
ios::nocreate	open fails if file does not exist
ios::noreplace	open fails if file already exists
ios::binary	open as a binary file

- A sequential file has to be accessed sequentially, to access the particular data in the file all the preceding data items have to be read and discarded.
- A random file allows access to the specific data without the need for accessing its preceding data items.
- The error-handling mechanisms of C++ is generally referred as exception handling.
- Exceptions are classified into synchronous and synchronous exceptions.
- The exceptions which occur during program execution, due to some fault in the input-data are known as synchronous exceptions.
- The exceptions caused by events external to the program are called asynchronous exceptions.
- The keywords which are used for exception handling are try,catch, throw.

EXERCISE – 1

1. In C++, a function contained within a class is called
   1. member function 2. class function 3. an operator 4. none
2. When a language has the capability to produce new data types, it is said to be
   1. reprehensible 2. encapsulated 3. abstracted 4. extensible
3. A normal C++ operator that acts in special ways on newly defined data types is said to be
   1. glorified 2. encapsulated 3. classified 4. overloaded
4. The && and | | operators
   1. compare numeric values 2. combine boolean values
   3. compare boolean values 4. combine numeric values
5. When an argument is passed by reference
   1. variable is created in the function to hold the argument's value
   2. the function cannot access the argument's value
   3. temporary variable is created in the calling program to hold the arguments value
   4. none
6. In a class specifier, data or functions designated private are accessible
   1. to any function in the program 2. only if u know the password
   3. to member functions of the class 4. only public members of the class
7. Operator overloading is
   1. making C++ operators work with objects 2. creating new C++ operators
   3. both 1 and 2 4. none
8. To convert from a user-defined class to a basic data type, you would most likely use
   1. a built-in conversion function 2. one-argument constructor
   3. an overloaded= operator 4. a conversion function that's member of the class
9. The scope resolution operator
   1. limits the visibility of variables to a certain function 2. resolves ambiguities
   3. has 3 : symbols 4. none
10. The new operator
    1. returns a pointer to a variable 2. creates a variable called new

3. tells the total memory available 4. sells the old things

11. A pure vitual function is a virtual function that
1. has no body 2. returns nothing 3. takes no arguments 4. all

12. The keyword typed if is used to
1.declare a member function that is defined in a subclass
2.designate the absence of a type
3.declare objects that can be modified outside of program control
4.declare a synonym of an existing type

13. The keyword asm is used to
1. specify a class declaration 2. specify a constant definition
3. allow information to be passed to assembler directly 4. none

14. Which is the declaration for a pointer to a function that returns an integer
1. int f() 2. int f() 3. int (*f) () 4. int * (*f)()

15. Which is the logical expression for n is even but not 8
1. (n%2==0) || (n!=8) 2. (n%2==0) && (n!=8) 3. (n<>8)&&(n%2) 4. none

16. Before the execution of the statement m*=n++; if the values of m and n are 5 and 2. The values after execution of the statement are
1. 3 15 2. 2 10 3. 3 10 4. 2 15

17. The symbol << is called
1. insertion operator 2. extraction operator 3. both 4. none

18. A class that contains atleast one pure virtual function is called as
1. pure class 2. abstract class 3. base class 4. derived class

19. What is the output of the following
```
cout.setf(ios::oct,ios::basefield);
cout<<30;
```
1. 30 2. 36 3. 24 4.error

20. In C++ , what is the output
```
void f(int a=1) { cout<<"hai";}
void f(void) { cout<<"rai"; }
void main(){ f0; }
```
1. hai 2. rai 3. error 4. none

21. Templates enables us to create a range of related
1. operators 2. lists 3. functions 4. none

22. In C++ , the structures are by default
1. public 2. private 3. protected 4.none

23. How many arguments can a copy constructor have
1. two 2. one 3. three 4. no limit

24. fstream class in C++ is
1.a predefined class to open a file in input mode only
2.a predefined class to open a file in output mode only
3.a predefined class to open a file in both input and output modes 4. none

25. In C++, suppose fp is a file stream for some file and is closed as
1. close(fp) 2. fp.close() 3. close 4. all

26. What is the output of the following C++ code

```
void main()
{
    cout.fill('*');
    cout.width(7);
    cout<<5250;
}
```

1.error 2. 5250 3. ***5250 4. 5250***

27. class A : public B , public C , public D { } is
1. an example of multilevel inheritance 2. an example of hybrid inheritance
3. an example of hierarchical inheritance 4. an example of multiple inheritance

28. The statements

```
int a =20;
cout<<"upper"<<a>>2<<"lower";
```

outputs
1. upper5lower 2. upper20lower 3. upper202lower 4. error

29. Which of the following is true
1.destructors can be overloaded but constructors cannot be overloaded
2.constructors can be overloaded but destructors cannot be overloaded
3.destructors can take arguments but constructors cannot 4.none

30. What is the output of the following program

```
int I=50;
void main()
{   int I =100;
    {
        int I =150;
        printf("%d %d", ::I,I);
    }
}
```

1. 50 100 2. 50 150 3. 50 100 4. 100 50

31. In which of the following cases inline expansion may not work
1. functions which contain static variables 2. functions which are recursive
3. both a and b 4. inline expansion always occurs

32. Which of the following is correct
1. ?: can be overloaded 2. :: can be overloaded
3. .* can be overloaded 3. new can be overloaded

33. A static function
1.should be called when an object is destroyed
2.is closely connected with an individual object of a class
3.can be called using the class name and function name
4.is used when a dummy object must be created

34. cout in C++ is
1. an object 2. a class 3. both a and b 4. none

35. cin in C++ can
1.read only one value at a time 2. read any number of values of same type only

3. read any number of values of same or different types 4. none

36. All the functions in abstract base class must be declared pure virtual
1. true 2. false 3. can't say 4. none

37. Which of the following properties about the static member variables is true
1.it has access to all members of its class
2.it has access to only static members of its class
3.it should be called only by using the object name
4. it can access only non-static members of its class

38. Which of the following is true about a static variable
1.it is initialized to 0 when the first object of its class is created
2.a separate copoy of the variables is created for each object
3.its value will be vanished after the termination of function
4.it is visible to all the classes in the program

39. Which of the following prototype is illegal
1. void prod(int a, int b, int c); 2. void prod(int a ,int b,int c=10);
3. void prod (int a=10, int b=3,int c); 4. void prod(int a=1,int b=4,int c=3);

40. In **private** derivation , which of the following is true
1. private members become private 2. public members become public
3. public members become private 4. protected members become public

41. An exception is caused by
1. hardware problem 2. a problem in operating system
3. syntax error 4. run-time error

42. Which of the following OOP feature is not supported by C++
1. polymorphism 2. delegation 3. persistance 4. genericity

43. The following is true about a pure OOP language
1.all the OOP features are implemented in it
2.every thing in it is in the form of classes and objects
3.C++ is a example of it
4.There will not be any impure classes and all the classes are filtered

44. Templates in C++ are used to support which OOP feature
1. encapsulation 2. delegation 3. genericity 4. inheritance

45. What is the output of the following program in C++

```
void main()
{
    int y=10;
    int &x=y;
    x*=(++y * 3);
    printf("%d",y);
}
```

1. 333 2. 363 3. 1089 4. none

46. Which of the following is false about inline functions
1. all the functions defined within the class are inline by default
2.inline functions are also called as open subroutines
3.they improve compilation speed

4.they may use the keyword inline

47. Which of the following is true
    1.overloading and overriding are the same
    2.overloaded functions may differ only in the return type
    3.overloaded functions should have same function name
    4.template functions cannot be overloaded
48. Which functions hinder the concept of data hiding
    1.virtual functions 2. template functions
    3. friend functions 4. overloaded functions
49. Which of the following is true about virtual functions
    1.they are imaginary functions and not real functions
    2.they are called as normal functions
    3.they are called by using pointers of base class
    4.they should always be pure without definition
50. Which of the following is true
    1.classes can be overloaded
    2.we can create any number of objects of a particular class
    3.there cannot be more than 20 objects of a particular class
    4.we can never access members by the class name

EXERCISE – 2

1. Which of the following is false
    1. a class can be left empty
    2. a class can be defined without a name
    3. a class with pure virtual function can have objects
    4.a class with pure virtual function can be used as base class
2. Which of the following is true about friendship in C++
    1. it is transitive 2. it is mutual
    3. should be used only in necessary 4. two data members can be close friends
3. Which of the following is true in C++
    1. exception handling is not supported
    2. there are some functions which are invoked automatically
    3. static data members can be left unintialized
    4.there can be more than one destructor in a class
4. Which of the following is true about constructors and destructors
    1. both of them can be virtual 2. both of them can be overloaded
    2. both of them can be in any number 4. both of them cannot return value
5. Which of the following is false
    1. classes can be nested 2. public part can be before private part
    3. classes can be self-referential 4. classes can be overloaded
6. Which of the following is false
    1. a class from which a live object is created should definetely have a constructor
    2. any member function of every object can access this pointer
    3.virual functions use static binding

4.data structures like trees, graphs and lists are implemented by self referential classes

7. The number of arguments in operator function for overloading a binary operator is
   1. one 2. two 3. three 4. none
8. Which type of inheritance has ambiguities and problems
   1. single inheritance 2. multiple inheritance
   3. hierarchical inheritance 4. multilevel inheritance
9. Which of the following is false
   1. constructors can not inherited from base class to derived class
   2. to create derived object both the constructors of base and derived are invoked
   3. first the constructor of the base class is invoked and then that ofderived
   4. first the destructor of the base class is invoked and then that of derived
10. What is the order of invokation of the constructors when object of class D is created , given class D : public A1,virtual A2
    1. D,A1,A2 2. D,A2,A1 3. A1,A2,D 4. A2,A1,D
11. To handle ambiguity caused due to multipath inheritance which is used
    1. virtual functions 2. templates 3. virtual base classes 4. friend functions
12. Which of the following is false about virtual functions
    1. they cannot be static members 2. they can be friend to other class
    3. a destructor can be virtual 4.they are accessed normally
13. Generic classes are nothing but
    1. container classes 2. template classes
    3. Empty classes 4. self-referential classes
14. Formatted I/O operations can be performed by
    1. ios stream calss member functions and flags 2. standard manipulators
    3. user-defined manipulators 4. all the above
15. Which of the following mode is used to open the file for writing
    1. ios::in 2. ios::out 3. ios::binary 4. ios::nocreate
16. Which of the following is not a key word related to exception handling
    1. try 2. throw 3. hit 4. catch
17. Memory leaks in C++ are less when compared to C
    1. yes 2. no 3. can't say 4. none
18. Which of the following is an example for a manipulator
    1. cin 2. auto 3. endl 4. cout
19. Can the memory allocated by new operator deallocated by using free() function
    1. yes 2. no 3. can't say 4. none
20. What is the syntax for deleting an array A dynamically
    1. delete A 2. delete A{} 3. delete [] A 4. delete [A]
21. The definition const int * xyz ; then xyz
    1.is a constant pointer 2.is a pointer constant
    3. both 4. none
22. Which of the following is an example of a pure object oriented language
    1. C 2. C++ 3. java 4. smalltalk

23. Which of the following operator has the highest precedence level
    1. :: 2. ++ 3. * 4. new
24. Which pointers cannot be dereferenced without explicit type casting
    1. constant pointers 2. null pointers 3.void pointers 4. Wild pointers
25. Can functions be defined in C++ structures
    1. yes 2. No 3. Can't say 4. None
26. The exceptions caused by some faults external to the program are called
    1. synchronous 2. Asynchronous 3. Either 4. Pure exceptions
27. C++ streams are treated as ___ because they have even the capability to change the data representation.
    1. Erasers 2. Filters 3. Browsers 4. Loaders
28. Can you use printf() statements and cout stream objects for producing the output in the same program
    1. yes 2. No 3. Cant say 4. None
29. In which of the following stream classes the operator << is overloaded
    1. istream 2. ostream 3. iostream 4. Ios
30. Which of the following is an example for a paramaterized manipulator in C++
    1. hex 2. endl 3. flush 4. Setw
31. What will be the output of the following code

```
#include <iostream.h>
void main()
{       cout.precision(2);
        cout<<7.002;
}
```

    1. 7.00 2. 7 3. 7.001 4. error
32. What is the output of the following code

```
#include <iostream.h>
void main()
{       int x=100;
        cout<<hex<<x<<oct<<(x>>2);
}
```

    1. 100 50 2. 64 25 3. 64 31 4. Error
33. Which of the following is not a ios flag
    1. Ios :: left 2. Ios:: right 3. Ios:: external 4. Ios::internal
34. What is the output of the following code

```
#include <iostream.h>
void main()
{       cout.put('B');
}
```

    1. B 2. 66 3. 98 4. Error
35. The minimum number of parameters required for defining an non-parameterized user-defined manipulator is
    1. 0 2. 1 3.2 4. 3

23. Which of the following operator has the highest precedence level
    1. :: 2. ++ 3. * 4. new
24. Which pointers cannot be dereferenced without explicit type casting
    1. constant pointers 2. null pointers 3.void pointers 4. Wild pointers
25. Can functions be defined in C++ structures
    1. yes 2. No 3. Can't say 4. None
26. The exceptions caused by some faults external to the program are called
    1. synchronous 2. Asynchronous 3. Either 4. Pure exceptions
27. C++ streams are treated as ___ because they have even the capability to change the data representation.
    1. Erasers 2. Filters 3. Browsers 4. Loaders
28. Can you use printf() statements and cout stream objects for producing the output in the same program
    1. yes 2. No 3. Cant say 4. None
29. In which of the following stream classes the operator << is overloaded
    1. istream 2. ostream 3. iostream 4. Ios
30. Which of the following is an example for a paramaterized manipulator in C++
    1. hex 2. endl 3. flush 4. Setw
31. What will be the output of the following code

```
#include <iostream.h>
void main()
{       cout.precision(2);
        cout<<7.002;
}
```

    1. 7.00 2. 7 3. 7.001 4. error
32. What is the output of the following code

```
#include <iostream.h>
void main()
{     int x=100;
      cout<<hex<<x<<oct<<(x>>2);
}
```

    1. 100 50 2. 64 25 3. 64 31 4. Error
33. Which of the following is not a ios flag
    1. Ios :: left 2. Ios:: right 3. Ios:: external 4. Ios::internal
34. What is the output of the following code

```
#include <iostream.h>
void main()
{       cout.put('B');
}
```

    1. B 2. 66 3. 98 4. Error
35. The minimum number of parameters required for defining an non-parameterized user-defined manipulator is
    1. 0 2. 1 3.2 4. 3

36. Which of the following is false about function templates
    1. they are also called generic functions
    2. they can be overloaded
    3. a function template can have multiple generic arguments
    4. a template function may not have any arguments
37. Which of the following is true about a template class
    1. all the member functions of a template class are treated as template functions
    2. a derived class of a template base class is also a template class
    3. the objects of template classes are not used in the same way as normal classes
    4. template classes cannot be inherited
38. Which of the following is not used for implementing polymorphism
    1. operator overloading 2. Function overloading
    3. inline functions 4. Virtual functions
39. If class B is derived from A and class C is derived from B then the pointer to the class A can
    1. point to only objects of class A 2. Point to only objects of A and B
    3. point to objects of A,B,C 4. Point to only objects of B and C
40. To get the maximum use of virtual functions they should be defined in
    1. Private part 2. Public part 3. Protected part 4. Any part
41. A derived class of an abstract class
    1. should be abstract 2. Should be concrete
    3. either abstract or concrete 4. Neither abstract nor concrete

42. Which of the following is true about virtual functions
    1. they improve execution speed 2. They degrade the execution speed
    3.sometimes improves and sometimes degrades
    4 . none
43. If x is a private data member of integer type of the class A. If abc is the object of the class A which invokes constructor which initialise x to 10. What is the output of the code

```
#include <iostream.h>
void main()
{       cout << abc.x;
}
```

    1. abc.x 2 .10 c. 0 4. error
44. The default visibility mode of inheritance is
    1. private 2. Public 3. protected 4. none
45. The type of inheritance in which a class is derived from another derived class is called
    1. multiple inheritance 2. Multi-level inheritance
    3. hierarchical inheritance 4. Multi-path inheritance
46. Which of the following is false
    1. a public member can access a private member of the same class
    2. a private member can access a public member of the same class
    3. a friend function can access private data of a particular class

4. protected members of a class cannot be acessed by private members

47. What is the order of invokation of the constructors when an object of class D is created given class D : public A, public C
1. D A C 2. A C D 3. C A D 4. A D C

48. The number of arguments for overloading an unary operator in operator function is
1. 0 2. 1 3.2 4.3

49. The conversion between objects of two user defined classes is done by using
1. one-argument constructor 2. Conversion function
3. either 1 or 2 d. done automatically by compiler

50. Operators can be overloaded by using even with these functions
1. inline functions 2. Friend functions 3. Overloaded functions 4. Virtual functions

51. Which of the following operators can be overloaded
1. :: 2. Sizeof 3. += 4. ?:

52. Which of the following function is called when an object goes out of scope
1. default constructor 2. Operator function 3. Destructor 4. Nothing

53. If an object is created by using the keyword const then such object is
1. live object 2. Name less object 3. Read only object 4. Pure object

54. Which of the following is false
1. data members can be initialized at the point of their definition
2. constructors can have default arguments
3. constant objects can be initialized by constructor
4. constructors defined in private part are useless

55. Following is not true about passing of objects as arguments
1. they can be passed by value 2. They can be passed by reference
3. they can be passed by auto_ptr 4. Objects can be passed

56. Assume class C with objects a,b,c. For the statement c=a-b to work correctly the overloaded operator – must
1. take two arguments 2. create named temporary object
3. Return value 4. Use the object of which it is a member as an operand

57. When you overload an arithmetic assignment operator, the result
1 .goes in the object to the right of the operator
2. goes in the object to the left of the operator
3. must be returned
4.goes in the object of which the operator is a member

58. If you want to sort many objects , it would be more efficient to
1. place them in an array and sort the array
2. place pointers to them in an array and sort the array
3. place them in a linked list and sort the linked list
4. both 1 and 2 are of equal efficiency

59. A static function
1. should be called when an object is destroyed
2. is closely connected with an individual object of a class
3. can be called using the class name and function name

4. is used when a dummy object must be created

60. The operation of assignment operator and that of the copy constructor are
    1. similar, except that the copy constructor creates a new object
    2. similar , except that the assignment operator copies member data
    3. different , except that they both create a new object
    4. none

61. A C++ stream is
    1. the flow of control through a function
    2. A flow of data from one place to another
    3. associated with a particular class 4. A file

62. We can output text to an object of class ofstream using the insertion operator << because
    1. the ofstream class is a stream
    2. The insertion operator works with all classes
    3.we are actually outputting to cout
    4. The insertion operator is overloaded in ofstream

63. The major goal of inheritance in C++ is
    1. to help modular programming 2. To facilitate the conversion of data types
    3. to facilitate reusability of code 4. To hide the details of base classes

64. In which of the following code fragments the variable x is evaluated to 8
    1. int x=30 x=x>>2 2. int x=33 x=x>>2
    3. int x=37 x=x>>2 4. Int x=18 x=x>>2

65. In C++ , a funtion can return multple values
    1. true 2. False 3. Some times false 4. None

66. Which of the following is a legal way to access a class data member using **this** pointer
    1. this->x 2. this.x 3. this.x 4.*(this.x)

67. The function show() is a member of the class A and obj is a object of A and ptr is a pointer to A . Which of the following are valid access statements
    1. *obj.show() 2. Obj->show() 3. Ptr->show 4. Ptr.show()

68. What is the output of the following program

```
#include <iostream.h>
#include <conio.h>
void main()
{   int number=-2;
    if(number>=0)
      if(number<0)
        cout<<"positive";
      else
        cout<<"negative";
}
```

    1. positive 2. Negative 3. Error 4. Nothing is printed

69. What is the output of the following code

```
#include <iostream.h>
void main()
```

```
{
  int x=10,y=15;
  x=((x<y)?(y+++x):(y---x));
  cout<<x<<" "<<y;
}
```

1. 25 16 2. 26 15 3. 10 16 4.error

70. What is the result of the expression (1 & 2) + (3 | 4) in base 10.

1. 1 2.2 3.8 4. 7

71. A class hierarchy

1. shows the same relationships as an organization chart
2. describes "has-a"relationship
3. describes "kind of" relationship
4. none

72. What is the output of the following code

```
#include <iostream.h>
void main()
{
  char k=65;
  cout<<++k<<endl;
}
```

1. 65 2. 66 3. A 4. B

KEY – EXERCISE – 1

1–1	2–4	3–4	4–2	5–1	6–3	7–1	8–4	9–2	10–1
11–1	12–4	13–3	14–3	15–2	16–3	17–1	18–2	19–2	20–3
21–3	22–1	23–2	24–3	25–2	26–3	27–4	28–1	29–2	30–2
31–3	32–4	33–3	34–1	35–3	36–2	37–2	38–1	39–3	40–3
41–4	42–3	43–2	44–3	45–2	46–3	47–3	48–3	49–3	50–2

KEY – EXERCISE – 2

1–3	2–3	3–2	4–4	5–4	6–4	7–1	8–2	9–4	10–4
11–3	12–4	13–2	14–4	15–2	16–3	17–1	18–3	19–2	20–3
21–2	22–4	23–1	24–3	25–1	26–2	27–2	28–2	29–2	30–4
31–2	32–3	33–3	34–1	35–2	36–4	37–1	38–3	39–3	40–2
41–3	42–2	43–4	44–1	45–2	46–4	47–2	48–1	49–3	50–2
51–3	52–3	53–3	54–1	55–1	56–4	57–2	58–2	59–3	60–1
61–2	62–4	63–3	64–2	65–2	66–1	67–3	68–4	69–1	70–4
71–3	72–4								