Uses of Destructor

- Destructor is used to free memory that is allocated through dynamic allocation.
- Destructor is used to perform house keeping operations.

Q; Describe the way to declare a template function as a friend of any class?

Template templatename
Class calssname
{
Friend void friend templatename (classname <templatename> astric const prt classname);
}

Q;two types of container?

Sequence and associative containers are collectively referred to as the first-class containers.

Q;State any two reasons why the virtual methods can not be static?

1-virtual method can not be static as it is dynamic
2-as virtual method is dynamic so it works automatically that is also another reason That virtual method can not be static.

Q;Explain the statement below,

vector<int> ivec(4, 3);

Ans; First vector instance to store integer values from which we need to find integer value.

vector<int>beyond(4),found(3);

beyond is initialized with first vector instance 4

Q;Explain two benefits of setter functions.

1- It minimize the changes to move the objects in inconsistent states
2- You can write checks in your setter functions to check the validity of data entered by the user, for example age functions to check to calculate the age from date entered.

Q;Consider the code below,

```
template< typename T >
class T1 {
```

```
public:
T i;
protected:
T j;
private:
T k;
friend void Test();
};
```

This code has a template class T1 with three members i,j and k and a friend function Test(), you have to describe which member/s of T1 will be available in function Test().

ANS;

```
public:
  T i;
  protected:
  T j;
```

Q;What do you mean by Stack unwinding?

ANS; When we want to check what happens actually to the local variables in the try block when then an exception is thrown this concept is called stack unwinding.

Q;What is random_iterator? What is relation between random_iterator and Vector?

Ans; Random_iterator: it provided both increment and decrement and also provide constant time methods for moving forward and backword in arbitrary sized steps. Ramdom iterator provide asentially all of the operations of ordinary c pointer arithmetic.

Vector class provide an stl style random access iterator for use with generic algorithm since neither the vactor nor the matrix classes are container classes in actuall. The iterator class is really an iterator of data object that is viewed by vector or matrix.

Q;What would be the output of this code?

```
class mother {
 public:
  mother ()
    { cout << "mother: no parameters\n"; }
  mother (int a)
    { cout << "mother: int parameter\n"; }
};

class daughter : public mother {
 public:
  daughter (int a)
    { cout << "daughter: int parameter\n\n"; }
};
```

```
class son : public mother {
  public:
    son (int a) : mother (a)
      { cout << "son: int parameter\n\n"; }
};

int main () {
  daughter rabia (0);
  son salman(0);
    return 0;
}
```

Ans; Output will be

Mother
Daughter: rabia
Son: salman

Q; Is Deque a Birectional Container?

ANS; Yes, deque behaves like queue (line) such that we can add elements on both sides of it.

Q;What is meant by Generic Programming?

ANS; Generic programming refers to programs containing generic abstractions general code that is same in logic for all data types like printArray function), then we instantiate that generic program abstraction (function, class) for a particular data type, such abstractions can work with many different types of data.

Q;Sort the following data in the order in which compiler searches a function?

Complete Specialization, Generic Template, Partial Specialization, Ordinary Function.
ANS; Specializations of this function template, instantiations with specific types, can be called just like an ordinary function:

```
cout << max(3, 7);   // outputs 7
```

The compiler examines the arguments used to call max and determines that this is a call to max(int, int). It then instantiates a version of the function where the parameterizing type T is int, making the equivalent of the following function:

```
int max(int x, int y)
{
   return x < y ? y : x;
}
```

the C++ Standard Template Library contains the function template max(x, y) which creates functions that return either x or y, whichever is larger. max() could be defined like this:

```
template <typename T>
T max(T x, T y)
{
   return x < y ? y : x;
```

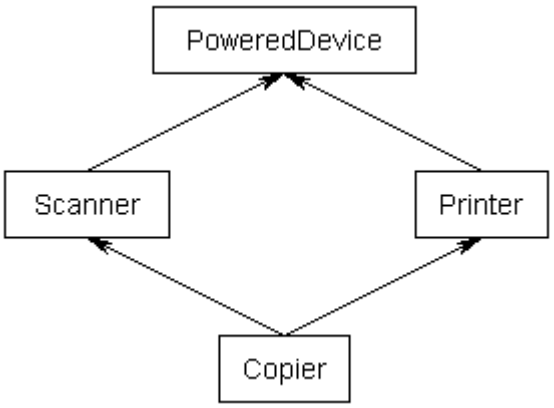
}

Q;State any conflict that may rise due to multiple inheritance?

ANS; The conflict may arise is the diamond problem, which our author likes to call the "diamond of doom". This occurs when a class multiply inherits from two classes which each inherit from a single base class. This leads to a diamond shaped inheritance pattern.For example, consider the following set of classes:

classPoweredDevice

{
};
classScanner: publicPoweredDevice
{
};
classPrinter: publicPoweredDevice
{
};
classCopier: publicScanner, publicPrinter
{
};



Scanners and printers are both powered devices, so they derived from PoweredDevice. However, a copy machine incorporates the functionality of both Scanners and Printers.Ambiguity also cause problem.

Q;Describe three properties necessary for a container to implement Generic Algorithms?

ANS; If you declare a container as holding pointers, you are responsible for managing the memory for the objects pointed to. The container classes will not automatically free memory for these objects when an item is erased from the container.

Container classes are expected to implement methods to do the following:

- create a new empty container (constructor),
- report the number of objects it stores (size),
- delete all the objects in the container (clear),
- insert new objects into the container,
- remove objects from it,
- provide access to the stored objects.

Q;Write three important features of virtual functions?

ANS; With virtual functions, derived classes can provide new implementations of functions from their base classes. When someone calls a virtual function of an object of the derived class, this new implementation is called, even if the caller uses a pointer to the base class, and doesn't even know about the particular derived class.
The virtual function is an option, and the language defaults to non virtual, which is the fastest configuration.
The derived class can completely "override" the implementation or "augment" it (by explicitly calling the base class implementation in addition to the new things it does).

Q; Given are two classes A and B. class B is inherited from class A. Write a code snippet(for main function) that polymorphically call the method of class B. Also what changes do you suggest in the given code segment that are required to call the class B method polymorphically.

```
class A
{
public:
void method() { cout<<"A's method \n"; }
};
class B : public A
{
public:
void method() { cout<<"B's method\n"; }
};
```

Ans:

```
public class Test
{
public class A {}
public class B extends A {}
private void test(A a)
{
System.out.println("test(A)");
}
private void test(B b)
{
System.out.println("test(B)");
}
```

```
public static void main(String[] args)
{
Test t = new Test();
A a = t.new A();
A b = t.new B();
t.test(a);
t.test(b);
}
}
```

Q. Write a detailed note on Exceptions in Destructors with the help of a coding example?

ANS;

Exceptions in Destructors:

An object is presumably created to do something. Some of the changes made by an object should persist after an object dies (is destructed) and some changes should not. Take an object implementing a SQL query. If a database field is updated via the SQL object then that change should persist after the SQL objects dies. To do its work the SQL object probably created a database connection and allocated a bunch of memory. When the SQL object dies we want to close the database connection and deallocate the memory, otherwise if a lot of SQL objects are created we will run out of database connections and/or memory.

The logic might look like:

```
Sql::~Sql()
{
  delete connection;
  delete buffer;
}
```

Let's say an exception is thrown while deleting the database connection. Will the buffer be deleted? No. Exceptions are basically non-local gotos with stack cleanup. The code for deleting the buffer will never be executed creating a gaping resource leak.

Special care must be taken to catch exceptions which may occur during object destruction. Special care must also be taken to fully destruct an object when it throws an exception.

Q;Write the syntax of declaring a pure virtual function in a class?

Ans:

Pure Virtual Function is a Virtual function with no body.

Declaration of Pure Virtual Function:

Since pure virtual function has no body, the programmer must add the notation =0 for declaration of the pure virtual function in the base class.

General Syntax of Pure Virtual Function takes the form:

```
class classname //This denotes the base class of C++ virtual function
{
public:
virtual void virtualfunctioname() = 0 //This denotes the pure virtual function in C++
```

};

Q;What is meant by direct base class ?

Ans; When a class-type is included in the class-base, it specifies the direct base class of the class being declared. If a class declaration has no class-base, or if the class-base lists only interface types, the direct base class is assumed to be object. A class inherits members from its direct base class, Deriving a class from more than one direct base class is called multiple inheritance.

Q;What is the purpose of template parameter?

Ans:

There are three kinds of template parameters:

• type
• non-type
• template

You can interchange the keywords class and typename in a template parameter declaration. You cannot use storage class specifiers (static and auto) in a template parameter declaration.

Q;Describe in simple words how we can use template specialization to enforce case sensitive specialization in String class.

Ans; The act of creating a new definition of a function, class, or member of a class from a template declaration and one or more template arguments is called template instantiation.The definition created from a template instantiation is called a specialization. A primary template is the template that is being specialized.create function objects to do the case-insensitive compares, and
then reuse them when also wanting to do case-insensitive sorting or searching.

Q;Can we use compiler generated default assignment operator in case our class is using dynamic memory? Justify your answer.

Ans:

the compiler does not make a separate copy of the object. Even if the types are not the same, the compiler is usually able to do a better job with initialization lists than with assignments.

Consider the following constructor that initializes member object x_ using an initialization list: square::square() : x_(whatever) { }. The most common benefit of doing this is improved performance. For example, if the expression whatever is the same type as member variable x_, the result of the whatever expression is constructed directly inside x_ — the compiler does not make a separate copy of the object. Even if the types are not the same, the compiler is usually able to do a better job with initialization lists than with assignments.

As if that wasn't bad enough, there's another source of inefficiency when using assignment in a constructor: the member object will get fully constructed by its default constructor, and this might, for example, allocate some default amount of memory or open some default file. All this work could be for naught if the whatever expression and/or assignment operator causes the object to close that file and/or release that memory

(e.g., if the default constructor didn't allocate a large enough pool of memory or if it opened the wrong file).

Q;Give the names of three ways to handle errors in a program.

Ans:

The function will throw DivideByZero as an exception that can then be caught by an exception-handling catch statement that catches exceptions of type int. The necessary construction for catching exceptions is a try catch system. If you wish to have your program check for exceptions, you must enclose the code that may have exceptions thrown in a try block.

The catch statement catches exceptions that are of the proper type. You can, for example, throw objects of a class to differentiate between several different exceptions. As well, once a catch statement is executed, the program continues to run from the end of the catch.

the errors can be handled outside of the regular code. This means that it is easier to structure the program code, and it makes dealing with errors more centralized. Finally, because the exception is passed back up the stack of calling functions, you can handle errors at any place you choose.

Q;Consider the following code,

```
class Base{
private:
void base1();
protected:
void base2();
public:
void base3();
};
class Derived: public Base{
private:
void derived1();
protected:
void derived2();
public:
void derived3();
};
int main(){
Derived * derived = new Derived();
return 0;
}
```

Fill the table below to tell which member functions of Base and Derived classes we can access using the Derived pointer in the code indicated in bold.

Ans:

Function Name Availability (Yes / No)?

base2() no

base3() yes

derived1() No

derived2() No

derived3() Yes

Q;What is the output produced by the following program?

```
#include<iostream.h>
void sample_function(double test) throw (int);
int main()
{
try
{
cout <<"Trying.\n";
sample_function(98.6);
cout << "Trying after call.\n";
}
catch(int)
{
cout << "Catching.\n";
}
cout << "End program.\n";
return 0;
}
void sample_function(double test) throw (int)
{
cout << "Starting sample_function.\n";
if(test < 100)
throw 42;
}
```

Ans:
Starting sample_function
Trying
Trying after call
Catching
End program

Templates

►In C++ generic programming is done using templates
►Two kinds
- Function Templates
- Class Templates

►Compiler generates different type-specific copies from a single template

Function Templates

►A function template can be parameterized to operate on different types of data

Association

- ► Objects in an object model interact with each other
- ► Usually an object provides services to several other objects

- ► An object keeps associations with other objects to delegate tasks

Kinds of Association

- ► Class Association
  - Inheritance
- ► Object Association
  - Simple Association
  - Composition
  - Aggregation

Simple Association

- ► Is the weakest link between objects
- ► Is a reference by which one object can interact with some other object
- ► Is simply called as “association”

Kinds of Simple Association

- ► w.r.t navigation
  - One-way Association
  - Two-way Association
- ► w.r.t Cardinality.
  - Binary Association
  - Ternary Association
  - N-ary Association

Q.give the diff b/w iterators and cursor?

Cursors

►A better way is to use cursors
►A cursor is a pointer that is declared outside the container / aggregate object
►Aggregate object provides methods that help a cursor to traverse the elements
- T* first()
- T* beyond()
- T* next(T*)

Iterators

►Iterator is an object that traverses a container without exposing its internal representation
►Iterators are for containers exactly like pointers are for ordinary data structures

Generic Iterators

►A generic iterator works with any kind of container
►To do so a generic iterator requires its container to provide three operations
- T* first()
- T* beyond()
- T* next(T*)

Advantages of iterators;

►With iterators more than one traversal can be pending on a single container
►Iterators allow to change the traversal strategy without changing the aggregate object

►They contribute towards data abstraction by emulating pointers

Q;Give the differences between virtual inheritance and multiple inheritance?

Virtual Inheritance

►Virtual inheritance must be used when necessary
►There are situation when programmer would want to use two distinct data members inherited from base class rather then one

Multiple Inheritance

►A class can inherit from more then one class
►The derived class inherits data members and functions form all the base classes
►Object of derived class can perform all the tasks that an object of base class can perform

Non-type Parameters

The parameters given in template definition other than those used for mentioning templates types are called Non-type Parameters.like
Template<class T,class U, int I> Here int I is Non-type Parameter.
►Template parameters may include non-type parameters
►The non-type parameters may have default values
►They are treated as constants
►Common use is static memory allocation

Object

An object has

- ► State (attributes)
- ► Well-defined behaviour (operations)
- ► Unique identity

Example – Ali is a Tangible Object

- ► State (attributes)
  - ▪ Name
  - ▪ Age
- ► behaviour (operations)
  - ▪ Walks
  - ▪ Eats
- ► Identity
  - ▪ His name

Syntax of nested try catch block?

```
int main( ) {
        try {
            try {
                throw 1;
            }
            catch( float ) { }
    }
     catch( int ) { }        }        return 0;        }
```