FINALTERM EXAMINATION

CS304- Object Oriented Programming (Session - 4)

Question No: 1 (Marks: 1) - Please choose one

Classes like TwoDimensionalShape and ThreeDimensionalShape would normally be concrete, while classes like Sphere and Cube would normally be abstract.

► True
► False

Question No: 2 (Marks: 1) - Please choose one

Virtual functions allow you to

► create an array of type pointer-to-base class that can hold pointers to derived classes.
► create functions that can never be accessed.
► group objects of different classes so they can all be accessed by the same function code.
► use the same function call to execute member functions of objects from different classes

Question No: 3 (Marks: 1) - Please choose one

A pointer to a base class can point to objects of a derived class.

► True
► False

Question No: 4 (Marks: 1) - Please choose one

A copy constructor is invoked when

► a function do not returns by value.
► an argument is passed by value.

► a function returns by reference.
► an argument is passed by reference.

Question No: 5 (Marks: 1) - Please choose one

Each try block can have _____ no. of catch blocks.

► 1
► 2
► 3
► As many as necessary.

Question No: 6 (Marks: 1) - Please choose one

Non Template Friend functions of a class are friends of _____instance/s of that class.

► All
► One specific
► All instances of one date type
► None of the given options

Question No: 7 (Marks: 1) - Please choose one

Template functions use _____ than ordinary functions.

► Greater Memory
► Lesser Memory
► Equal Memory
► None of the given options

Question No: 8 (Marks: 1) - Please choose one

The find() algorithm

► finds matching sequences of elements in two containers.
► finds a container that matches a specified container.
► takes iterators as its first two arguments.
► takes container elements as its first two arguments.

Question No: 9 (Marks: 1) - Please choose one

The copy() algorithm returns an iterator to

► the last element copied from.
► the last element copied to.
► the element one past the last element copied from.
► the element one past the last element copied to.

Question No: 10 (Marks: 1) - Please choose one

If you define a vector v with the default constructor, and define another vector w with a one-argument constructor to a size of 11, and insert 3 elements into each of these vectors with push_back(), then the size() member function will return _____ for v and _____ for w.

► 11 for v and 3 for w.

► 0 for v and 0 for w.
► 0 for v and 3 for w.
► 3 for v and 11 for w.

Question No: 11 (Marks: 1) - Please choose one

Which is not the Advantage of inheritance?

► providing class growth through natural selection.
► facilitating class libraries.
► avoiding the rewriting of code.
► providing a useful conceptual framework.

Question No: 12 (Marks: 1) - Please choose one

```
class DocElement
{
public:
    virtual void Print() { cout << "Generic element"; }
};
class Heading : public DocElement
{
public:
    void Print() { cout << "Heading element"; }
};
```

```
class Paragraph : public DocElement
{
public:
    void Print() { cout << "Paragraph element"; }
};
void main()
{
    DocElement * p = new Paragraph();

    p->Print();
}
```

When you run this program, it will print out a single line to the console output.

What will be in that line?

Select one correct answer from the following list:

- ► Generic element
- ► Heading element
- ► Paragraph element
- ► Nothing will be printed.

Question No: 13 (Marks: 1) - Please choose one

Which type of inheritance is being represented by the following statement,

```
class X : public A, public B { ... ... };
```

- ► Single inheritance
- ► Multiple inheritance
- ► Double inheritance
- ► None of the given options

Question No: 14 (Marks: 1) - Please choose one

When we write a class template the first line must be:

► template < class *class_name*>
► template < class *data_type*>
► template < class T >

Here T can be replaced with any name but it is preferable.

► class class-name()

class template<*class_name*>

Question No: 15 (Marks: 1) - Please choose one

Function templates should be used where code and behavior must be identical.

► True
► False

Question No: 16 (Marks: 1) - Please choose one

Which of the following is/are advantage[s] of generic programming?

► Reusability
► Writability
► Maintainability
► All of given

Question No: 17 (Marks: 1) - Please choose one

The specialization pattern <T*> after the name says that this specialization is to be used for every,

► data type
► meta type

► virtual type

► pointer type

Question No: 18 (Marks: 1) - Please choose one

A range is often supplied to an algorithm by two _____ values.

► italic
► iteration
► iterator
► None of given

Question No: 19 (Marks: 1) - Please choose one

Which of the following is an integral part of an object?

► State

► Behavior
► Unique identity
► All of the given

Question No: 20 (Marks: 1) - Please choose one

Consider the following statement

Cupboard has books

What is the relationship between Cupboard and books?

► Composition
► Aggregation
► Inheritance
► None of the given options

Question No: 21 (Marks: 1) - Please choose one

Which sentence clearly defines an object?

► one instance of a class.
► another word for a class.
► a class with static methods.
► a method that accesses class attributes.

Question No: 22 (Marks: 1) - Please choose one

_____, which means if A declares B as its friend it does NOT mean that A can access private data of B. It only means that B can access all data of A.

► Friendship is one way only
► Friendship is two way only
► NO Friendship between classes
► Any kind of friendship

Question No: 23 (Marks: 1) - Please choose one

The statement objA=objB; will cause a compiler error if the objects are of different classes.

► True
► False

Question No: 24 (Marks: 1) - Please choose one

Consider the call given below of an overloaded operator "+",
Rational_number_1 + Rational_number_2
Where Rational_number_1 and Rational_number_2 are the two objects of Rational_number class (a user defined class). Identify which of the above two objects will be passed as an argument to the overloaded operator function?

- ► Rational_number_1
- ► Rational_number_2
- ► Both Rational_number_1 & Rational_number_2
- ► any of the two objects, randomly

Question No: 25 (Marks: 1) - Please choose one

If a class D has been derived using protected inheritance from class B (If B is a protected base and D is derived class) then public and protected members of B -------- accessed by member functions and friends of class D and classes derived from D

- ► can be
- ► cannot be
- ► does restirct to be
- ► not given

Question No: 26 (Marks: 1) - Please choose one

In Private ---------- only member functions and friend classes or functions of a derived class can convert pointer or reference of derived object to that of parent object

- ► specialization
- ► inheritance
- ► abstraction
- ► composition

Question No: 27 (Marks: 2)

Give two uses of a destructor.

Question No: 28 (Marks: 2)

Describe the way to declare a template class as a friend class of any other class.

Question No: 29 (Marks: 2)

Give the name of two basic types of containers collectively called First class containers?

Question No: 30 (Marks: 2)

State any conflict that may rise due to multiple inheritance?

Question No: 31 (Marks: 3)

What will be the output after executing the following code?

```
class c1{
public:
virtual void function(){
cout<<"I am in c1"<<endl;
```

```
}


};
class c2: public c1{
public:
void function(){
cout<<"I am in c2"<<endl;

}

};
class c3: public c1 {
public:
void function(){
cout<<"I am in c3"<<endl;
}

};

int main(){

c1 * test1 = new c2();
c1 * test2 = new c3();
test1->function();
test2->function();
system("PAUSE");
return 0;
}
```

Question No: 32 (Marks: 3)

If we declare a function as friend of a template class will it be a friend for a particular data type or for all data types of that class.

Question No: 33 (Marks: 3)

Tell the logical error/s in the code given below with reference to resource management; also describe how we can correct that error/s.

```
class Test{

public:
int function1(){
                try{
```

```
                    FILE *fileptr = fopen("filename.txt","w");
                     throw exception();
                     fclose(fileptr);
                     return 0;
                     }
                     catch(Exception e){
                     ...
                     }
}
};
```

Question No: 34 (Marks: 5)

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;
}
```

Question No: 35 (Marks: 5)

The code given below has one template function as a friend of a template class,

1. You have to identify any error/s in this code and describe the reason for error/s.
2. Give the correct code after removing the error/s.

```
template<typename U>
void Test(U);
template< class T >

class B {
        int data;
        public:
        friend void Test<>( T );
};

template<typename U>
void Test(U u){
   B < int> b1;
   b1.data = 7;
   }
int main(int argc, char *argv[])
{
   char i;
   Test(i);
   system("PAUSE");
   return 0;
}
```

Question No: 36 (Marks: 5)

Consider the following class,

```
class Base
{
        char * p;
public:
        Base() { p = new char[10]; }

        ~Base() { delete [] p; }
};
class Derived : public Base
{
        char * q;
public:
        Derived() { q = new char[20]; }

        ~Derived() { delete [] q; }
};
void foo()
{
        Base* p = new Derived();
```

```
        delete p;
}
```

With this program, every time function foo is called, some memory will leak.
Explain why memory will leak. Also, explain how to fix this problem.

FINALTERM EXAMINATION
Fall 2009
CS304- Object Oriented Programming (Session - 1)

Time: 120 min
Marks: 75

Question No: 1 (Marks: 1) - Please choose one

Which one of the following terms must relate to **polymorphism?**

- ► Static allocation
- ► Static typing
- ► **Dynamic binding**
- ► Dynamic allocation

Question No: 2 (Marks: 1) - Please choose one

Multiple inheritance can be of type

- ► Public
- ► Private

► Protected
► **All of the given**

Question No: 3 (Marks: 1) - Please choose one

When a subclass specifies an alternative definition for an attribute or method of its superclass, it is _____ the definition in the superclass.

► overload
► **overriding**
► copy riding
► none of given

Question No: 4 (Marks: 1) - Please choose one

Like template functions, a class template may not handle all the types successfully.

► **True**
► False

Question No: 5 (Marks: 1) - Please choose one

It is sometimes useful to specify a class from which no objects will ever be created.

► True
► **False**

Question No: 6 (Marks: 1) - Please choose one

Assume a class Derv that is privately derived from class Base. An object of class Derv located in main() can access

► public members of Derv.
► protected members of Derv.

► private members of Derv.
► **protected members of Base.**

Question No: 7 (Marks: 1) - Please choose one

A pointer to a base class can point to objects of a derived class.

► **True**
► False

Question No: 8 (Marks: 1) - Please choose one

A copy constructor is invoked when

► a function do not returns by value.
► an argument is passed by value.
► **a function returns by reference.**
► an argument is passed by reference.

Question No: 9 (Marks: 1) - Please choose one

A function call is resolved at run-time in_____

► non-virtual member function.
► **virtual member function.**
► Both non-virtual member and virtual member function.
► None of given

Question No: 10 (Marks: 1) - Please choose one

When the base class and the derived class have a member function with the same name, you must be more specific which function you want to call (using _____).

- **scope resolution operator**
- dot operator
- null operator
- Operator overloading

Question No: 11 (Marks: 1) - Please choose one

Each try block can have _____ no. of catch blocks.

- 1
- 2
- 3
- **As many as necessary.**

Question No: 12 (Marks: 1) - Please choose one

Two important STL associative containers are _____ and _____.

- set,map
- **sequence,mapping**
- setmet,multipule
- sit,mat

Question No: 13 (Marks: 1) - Please choose one

The mechanism of selecting function at run time according to the nature of calling object is called,

- late binding
- static binding
- virtual binding
- **None of the given options**

Question No: 14 (Marks: 1) - Please choose one

An abstract class is useful when,

- We do not derive any class from it.
- There are multiple paths from one derived class to another.
- **We do not want to instantiate its object.**
- You want to defer the declaration of the class.

Question No: 15 (Marks: 1) - Please choose one

Which of the following is incorrect line regarding function template?

- template<class T>
- template <typename U>
- **Class<template T>**
- template < class T, class U>

Question No: 16 (Marks: 1) - Please choose one

Which of the following is/are advantage[s] of generic programming?

- Reusability
- Writability
- Maintainability
- **All of given**

Question No: 17 (Marks: 1) - Please choose one

By default the vector data items are initialized to ____

- **0**
- 0.0
- 1
- null

Question No: 18 (Marks: 1) - Please choose one

Which one of the following functions returns the total number of elements in a vector.

- ► length();
- ► **size();**
- ► ele();
- ► veclen();

Question No: 19 (Marks: 1) - Please choose one

Suppose you create an uninitialized vector as follows:

vector<int> evec;
After adding the statment,
evec.push_back(21);
what will happen?

- ► The following statement will add an element to the start (the back) of evec and will initialize it with the value 21.
- ► The following statement will add an element to the center of evec and will reinitialize it with the value 21.
- ► The following statement will delete an element to the end (the back) of evec and will reinitialize it with the value 21.
- ► **The following statement will add an element to the end (the back) of evec and initialize it with the value 21.**

Question No: 20 (Marks: 1) - Please choose one

An STL container can not be used to,

- ► hold objects of class employee.
- ► store elements in a way that makes them quickly accessible.
- ► **compile c++ programs.**
- ► organize the way objects are stored in memory

Question No: 21 (Marks: 1) - Please choose one

Algorithms can only be implemented using STL containers.

- ► True
- ► **False**

Question No: 22 (Marks: 1) - Please choose one

The main function of scope resolution operator (**::**) is,

► **To define an object**
► To define a data member
► To link the definition of an identifier to its declaration
► To make a class private

Question No: 23 (Marks: 1) - Please choose one

When is a constructor called?

► Each time the constructor identifier is used in a program statement
► **During the instantiation of a new object**
► During the construction of a new class
► At the beginning of any program execution

Question No: 24 (Marks: 1) - Please choose one

Consider the code below,

```
class Fred {
public:
Fred();
...
};
int main()
{
Fred a[10];
Fred* p = new Fred[10];
...
}
```

Select the best option,

► Fred a[10]; calls the default constructor 09 times
Fred* p = new Fred[10]; calls the default constructor 10 times

► **Produce an error**

► Fred a[10]; calls the default constructor 11 times
Fred* p = new Fred[10]; calls the default constructor 11 times

► Fred a[10]; calls the default constructor 10 times
Fred* p = new Fred[10]; calls the default constructor 10 times

Question No: 25 (Marks: 1) - Please choose one

Associativity can be changed in operator overloading.

- ► True
- ► **False**

Question No: 26 (Marks: 1) - Please choose one

A normal C++ operator that acts in special ways on newly defined data types is said to be

- ► glorified.
- ► encapsulated.
- ► **classified.**
- ► overloaded.

Question No: 27 (Marks: 1) - Please choose one

Which operator can not be overloaded?

- ► The relation operator (>=)
- ► Assignment operator (=)
- ► Script operator ([])
- ► **Conditional operator (? :)**

Question No: 28 (Marks: 1) - Please choose one

Suppose obj1 and obj2 are two objects of a user defined class A. An + operator is overloaded to add obj1 and obj2 using the function call obj1+obj2.
Identify the correct function prototype against the given call?

- A operator + (A &obj);
- int + operator();
- **int operator (plus) ();**
- A operator(A &obj3);

Question No: 29 (Marks: 1) - Please choose one

Default constructor is such constructor which either has no ---------or if it has some parameters these have -------- values

- Parameter, temporary
- Null, Parameter
- **Parameter, default**
- non of the given

Question No: 30 (Marks: 1) - Please choose one

Public methods of base class can --------- be accessed in its derived class

- directly
- **inderectly**
- simultaniously
- non of the given

Question No: 31 (Marks: 1)

Is **Deque** a Birectional Container?

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

Question No: 32 (Marks: 1)

What is meant by Generic Programming?

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.

Question No: 33 (Marks: 2)

Sort the following data in the order in which compiler searches a function?
Complete Specialization, Generic Template, Partial Specialization, Ordinary Function.

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;
}
```

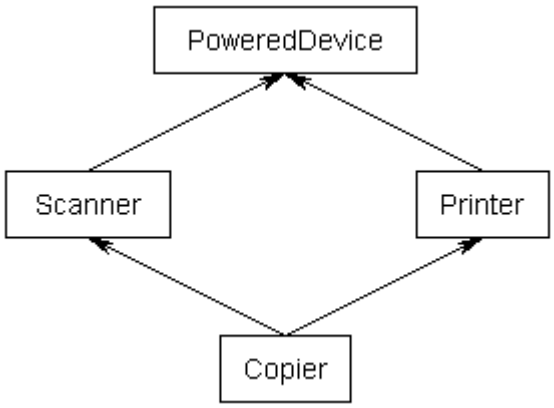
Question No: 34 (Marks: 2)

State any conflict that may rise due to multiple inheritance?

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.

Question No: 35 (Marks: 3)

Describe three properties necessary for a container to implement Generic Algorithms.

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.

Question No: 36 (Marks: 3)

Write three important features of virtual functions.

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).

Question No: 37 (Marks: 3)

Consider the code below,

```
#include <iostream>
#include <stdlib.h>
using namespace std;
class Shape{
        public:
        void Draw(){cout<<"shape"<<endl;}
    };
class Line : public Shape{
        public:
        void Draw(){cout<<"Line"<<endl;}
        };
class Circle : public Shape{
        public:
        void Draw(){cout<<"Circle"<<endl;}
        };
int main(int argc, char *argv[])
{
  Shape * ptr1 = new Shape();
  Shape * ptr2 = new Line();
  Shape * ptr3 = new Circle();

  ptr1->Draw();
  ptr2->Draw();
  ptr3->Draw();
  system("PAUSE");
  return 0;
}
```

This code shows output,

Shape
Shape
Shape

Give the reason for this output

Suppose we want to show the output,

Shape
Line
Circle

How we can change the code to do that?

```
class shape { public:
  void draw();
};
class circle : public shape { };
int main(int argc, char **argv){
  circle my_circle;
  my_circle.draw();
          }
```

While this has all the usual advantages, e.g., code reuse, the real power of polymorphism comes into play when draw is declared to be virtual or pure virtual, as follows:

```
class shape{ public:
  virtual void draw()=0;
};
class circle : public shape { public:
  void draw();
       }
```

Here, circle has declared its own draw function, which can define behavior appropriate for a circle. Similarly, we could define other classes derived from shape, which provide their own versions of draw. Now, because all the classes implement the shape interface, we can create collections of objects that can provide different behavior invoked in a consistent manner (calling the draw member function). An example of this is shown here.

```
shape *shape_list[3];   // the array that will
                        // pointer to our shape objects
shape[0] = new shape;  // three types of shapes
shape[1] = new line;  // we have defined
shape[2] = new circle;
for(int i = 0; i < 3; i++){
  shape_list[i].draw();
            }
```

When we invoke the draw function for each object on the list, we do not need to know anything about each object; C++ handles the details of invoking the correct version of draw. This is a very powerful technique, allowing us to provide extensibility in our designs. Now we can add new classes derived from shape to provide whatever behavior we desire. The key here is that we have separated the interface (the prototype for shape) from the implementation.

Question No: 38 (Marks: 5)

There are some errors in the code given below, you have to

1. Indicate the line no. with error/s
2. Give the reason for error/s
3. Correct the error/s.

```
1. #include <iostream>          this will be #include <iostream.h>
2. #include <stdlib.h>

3. using namespace std;
4. template <typename T>
5. class MyClass{
6. public:
7. MyClass(){
8. cout<<"This is class1"<<endl;
9.  }
10. };
11. template <typename T>
12. class MyClass<int*>{
13. public:
14. MyClass(){
15. cout<<"This is class2"<<endl;
16. }
17. };
18. int main(int argc, char *argv[])
19. {
20. MyClass<int> c1;
21. MyClass<int*> c2;
22. system("PAUSE");
23. return 0;
24. }
```

Question No: 39 (Marks: 5)

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);
}
}
```

Question No: 40 (Marks: 10)

Create built-in STL (Standard Template Library) **vector class object** for **strings** and add in it some words by taking input from user, then apply the sort() algorithm to array of words stored in this vector class object.
Hint:Use push_back() to add the words in vector class object, and the [] operator and size() to display these sorted words.

The STL is the containers, iterators and algorithms component of the proposed C++ Standard Library [ANSI95]. It represents a novel application of principles which have their roots in styles of programming other than Object-orientation.

```
void listWords(istream& in, ostream& out)
```

```
{
        string s;

        while (!in.eof() && in >> s) {
                add s to some container
        }

        sort the strings in the container
        remove the duplicates

        for (each string t in container) {
                out << t;
        }
}
```

For now, assume that a word is defined as a whitespace-separated string as delivered by the stream extraction operator. Later on we will consider ways of refining this definition. Given the way this problem is expressed, we can implement this program directly, if naïvely. The STL container class vector will suffice to hold the words: applying the algorithms sort and unique provides the required result.

```
void listWords(istream& in, ostream& out)
{
        string s;
        vector<string> v;

        while (!in.eof() && in >> s)
                v.push_back(s);                 // (1)

        sort(v.begin(), v.end());

        vector<string>::iterator e
                = unique(v.begin(), v.end());   // (2)

        for (vector<string>::iterator b = v.begin();
                b != e;
                b++) {
                out << *b << endl;
        }
}
```

At (1) the vector member function push_back() is used to add to the end of the vector. This can also be done using the insert member, which takes as a parameter an iterator identifying the position in the vector at which to place the added element:

```
        v.insert(v.end(), s);
```

This allows us to add at any position in the vector. Be aware, though, that adding anywhere other than the end implies the overhead of physically shifting all elements from the insertion point to the end to make room for the new value. For this reason, and given the choices made in this example, attempts to optimise this code by maintaining the

vector in sorted order are unwise. Replace vector with list and this becomes possible - although in both cases a search over the container will be necessary to determine the correct position of insertion.
The unique algorithm has the surprising property of not changing the length of the container to which it is applied (it can hardly do this, as it has access not to the underlying container, but only to the pair of iterators it is passed). Instead, it guarantees that duplicates are removed by moving unique entries towards the beginning of the container, returning an iterator indicating the new end of the container. This can be used directly (as here, at (2)), conversely it can be passed to the erase member with the old end iterator, to truncate the container.

Question No: 41 (Marks: 10)

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

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.

Example code for exception ……

```
#include<iostream.h>
#include<conio.c>
class Exception {
private:
```

```
 char message[30]  ;
public:

 Exception() {strcpy(message,"There is not enough stock");}
 char * get_message() { return message; }
};
class Item {
private:

 int stock ;
 int required_quantity;
public:

 Item(int stk, int qty)
    {
     stock =  stk;
     required_quantity =  qty;
 }
    int get_stock()
    {
       return stock;
    }

    int get_required_quantity()
    {
       return required_quantity;
    }

    void order()
    {
        if (get_stock()< get_required_quantity())

        throw Exception();
                  else
                  cout<<"The required quantity of item is available in the stock";
    }

     ~Item(){}
};

void main()
{

    Item obj(10, 20);

 try
```

```
  {
obj.order();
}
catch(Exception & exp2)
  {
   getch();
cout << "Exception: " << exp2.get_message() << endl;
}
getch();
```

FINALTERM EXAMINATION
Fall 2009
CS304- Object Oriented Programming (Session - 4)

Ref No: 1130772
Time: 120 min
Marks: 75

Question No: 1 (Marks: 1) - Please choose one

A template provides a convenient way to make a family of

► **variables and data members**

► functions and classes

► classes and exceptions

► programs and algorithms

Question No: 2 (Marks: 1) - Please choose one

Which one of the following terms must relate to **polymorphism?**

► Static allocation

► Static typing
► **Dynamic binding**
► Dynamic allocation

Question No: 3 (Marks: 1) - Please choose one

What is true about function templates?

- ▶ The compiler generates only one copy of the function template
- ▶ **The compiler generates a copy of function respective to each type of data**
- ▶ The compiler can only generate copy for the int type data
- ▶ None of the given.

Question No: 4 (Marks: 1) - Please choose one

Which of the following is the best approach if it is required to have more than one functions having exactly same functionality and implemented on different data types?

- ▶ **Templates**
- ▶ Overloading
- ▶ Data hiding
- ▶ Encapsulation

Question No: 5 (Marks: 1) - Please choose one

```
template <>
class Vector<char*> { }
```

This is an example of partial specialization.

- ▶ True
- ▶ **False**

Question No: 6 (Marks: 1) - Please choose one

Classes like TwoDimensionalShape and ThreeDimensionalShape would normally be concrete, while classes like Sphere and Cube would normally be abstract.

- ▶ **True**
- ▶ False

Question No: 7 (Marks: 1) - Please choose one

A non-virtual member function is defined in a base class and overridden in a derived class; if that function is called through a base-class pointer to a derived class object, the derived-class version is used.

► **True**

► False

Question No: 8 (Marks: 1) - Please choose one

Assume a class Derv that is privately derived from class Base. An object of class Derv located in main() can access

► public members of Derv.
► **protected members of Derv.**
► private members of Derv.
► protected members of Base.

Question No: 9 (Marks: 1) - Please choose one

In order to define a class template, the first line of definition must be:

► **template <typename T>**
► typename <template T>
► Template Class <ClassName>
► Class <Template T>

Question No: 10 (Marks: 1) - Please choose one

If there is a pointer p to objects of a base class, and it contains the address of an object of a derived class, and both classes contain a nonvirtual member function, ding(), then the statement p->ding(); will cause the version of ding() in the _____ class to be executed.

► Base
► Derived

- Abstract
- virtual

Question No: 11 (Marks: 1) - Please choose one

When the base class and the derived class have a member function with the same name, you must be more specific which function you want to call (using _____).

- scope resolution operator
- dot operator
- **null operator**
- Operator overloading

Question No: 12 (Marks: 1) - Please choose one

Non Template Friend functions of a class are friends of _____instance/s of that class.

- All
- **One specific**
- All instances of one date type
- None of the given options

Question No: 13 (Marks: 1) - Please choose one

The find() algorithm

- finds matching sequences of elements in two containers.
- **finds a container that matches a specified container.**
- takes iterators as its first two arguments.
- takes container elements as its first two arguments.

Question No: 14 (Marks: 1) - Please choose one

If you define a vector v with the default constructor, and define another vector w with a one-argument constructor to a size of 11, and insert 3 elements into each of these vectors with push_back(), then the size() member function will return _____ for v and _____ for w.

- 11 for v and 3 for w.

► 0 for v and 0 for w.
► 0 for v and 3 for w.
► **3 for v and 11 for w.**

Question No: 15 (Marks: 1) - Please choose one

Which of the following may not be an integral part of an object?

► State
► Behavior
► Protected data members
► **All of given**

Question No: 16 (Marks: 1) - Please choose one

Which is not the Advantage of inheritance?

► providing class growth through natural selection.
► **facilitating class libraries.**
► avoiding the rewriting of code.
► providing a useful conceptual framework.

Question No: 17 (Marks: 1) - Please choose one

```
class DocElement
{
public:
    virtual void Print() { cout << "Generic element"; }
};
class Heading : public DocElement
{
public:
    void Print() { cout << "Heading element"; }
};
class Paragraph : public DocElement
{
public:
    void Print() { cout << "Paragraph element"; }
};
```

```
void main()
{
    DocElement * p = new Paragraph();

    p->Print();
}
```

When you run this program, it will print out a single line to the console output.

What will be in that line?

Select one correct answer from the following list:

- ► Generic element
- ► Heading element
- ► **Paragraph element**
- ► Nothing will be printed.

Question No: 18 (Marks: 1) - Please choose one

When a virtual function is called by referencing a specific object by name and using the dot member selection operator (e.g., squareObject.draw()), the reference is resolved at compile time.

- ► **True**
- ► False

Question No: 19 (Marks: 1) - Please choose one

In case of multiple inheritance a derived class inherits,

- ► Only the public member functions of its base classes
- ► Only the public data members of its base classes
- ► **Both public data members and member functions of all its base classes**
- ► Data members and member functions of any two base classes

Question No: 20 (Marks: 1) - Please choose one

When we write a class template the first line must be:

- ▶ template < class *class_name*>
- ▶ template < class *data_type*>
- ▶ **template < class T >**

Here T can be replaced with any name but it is preferable.

- ▶ class class-name()

class template<*class_name*>

Question No: 21 (Marks: 1) - Please choose one

Which of the following is incorrect line regarding function template?

- ▶ template<class T>
- ▶ template <typename U>
- ▶ **Class<template T>**
- ▶ template < class T, class U>

Question No: 22 (Marks: 1) - Please choose one

An STL container can not be used to,

- ▶ hold objects of class employee.
- ▶ store elements in a way that makes them quickly accessible.
- ▶ compile c++ programs.
- ▶ organize the way objects are stored in memory

Question No: 23 (Marks: 1) - Please choose one

Algorithms can only be implemented using STL containers.

- ▶ **True**
- ▶ False

Question No: 24 (Marks: 1) - Please choose one

Consider a class named Vehicle, which of the following can be the instance of class Vehicle?

1. Car
2. Computer
3. Desk
4. Ahmed
5. Bicycle
6. Truck

► 1, 4, 5

► 2, 5, 6

► 1, 2, 3, 6

► **1, 5, 6**

Question No: 25 (Marks: 1) - Please choose one

Consider the code below,

```
class Fred {
public:
Fred();
...
};
int main()
{
Fred a[10];
Fred* p = new Fred[10];
...
}
```

Select the best option,

► Fred a[10]; calls the default constructor 09 times
Fred* p = new Fred[10]; calls the default constructor 10 times

► Produce an error

► **Fred a[10]; calls the default constructor 11 times**
Fred* p = new Fred[10]; calls the default constructor 11 times

► Fred a[10]; calls the default constructor 10 times
Fred* p = new Fred[10]; calls the default constructor 10 times

Question No: 26 (Marks: 1) - Please choose one

When a variable is define as **static** in a class then all object of this class,

- Have different copies of this variable
- **Have same copy of this variable**
- Can not access this variable
- None of given

Question No: 27 (Marks: 1) - Please choose one

The life of sub object is dependant on the life of master class in _____.

- Separation
- **Composition**
- Aggregation
- None of the given

Question No: 28 (Marks: 1) - Please choose one

_____, which means if A declares B as its friend it does NOT mean that A can access private data of B. It only means that B can access all data of A.

- Friendship is one way only
- Friendship is two way only
- NO Friendship between classes
- Any kind of friendship

Question No: 29 (Marks: 1) - Please choose one

Which of the following operators always takes no argument if overloaded?

- /
- -
- +
- ++

Question No: 30 (Marks: 1) - Please choose one

In Private ---------- only member functions and friend classes or functions of a derived class can convert pointer or reference of derived object to that of parent object

- ► specialization
- ► inheritance
- ► abstraction
- ► composition

Question No: 31 (Marks: 1)

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++
};
```

Question No: 32 (Marks: 1)

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.

Question No: 33 (Marks: 2)

Describe the way to declare a template class as a friend class of any other class.

Ans
The following example is use of a class template:

```
template<class L> class Key
                {
   L k;
   L* kptr;
   int length;
public:
   Key(L);
   // ...
};
```

Suppose the following declarations appear later:

```
Key<int> i;
 Key<char*> c;
 Key<mytype> m;
```

The compiler would create three objects.

Question No: 34 (Marks: 2)

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.

Question No: 35 (Marks: 3)

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.

Question No: 36 (Marks: 3)

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).

Question No: 37 (Marks: 3)

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.

Question No: 38 (Marks: 5)

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

Question No: 39 (Marks: 5)

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

Question No: 40 (Marks: 10)

Write a publicly derived class "E**mployee"** that is derived from base class named "**Company".** Both classes will have function **"create()"**. Make virtual function of base class and override same function in derived class. Function create will have an output statement of your own choice.

In "**main**" Create an object of base class and call both functions with same object type.

Question No: 41 (Marks: 10)

Write a program in C++ which creates three classes named as

1. **Equation**
2. **Linear**
3. **Quadratic**

Where Linear and Quadratic are inherited from Equation

Each class has the method Graph. Graph method should be pure virtual in Equation class.

This method should be overridden in both the inherited classes. It is meant to display the Graph shape of its respective class. Graph method of Linear will display the message;

Straight line

Similarly, the Graph method of Quadratic will display the message;

Parabola

In main, call the Graph method of both the Linear and Quadratic equations polymorphically through the parent class (Equation).

Ans:

```
#include "fraction.h"
#include <iostream>
#include <string>
#include <string.h>
#include <stdlib.h>
class equation;

class equation {
   int a, b;
  public:
   int c ()
     {return (c);}
   void convert (Cequation);
};

class linear {
  private:
   int side;
  public:
   void set_side (int a)
     {side=a;}
   friend class equation;
};
```

```
void equation::convert (Cequation) {
  a = 23;
  b = 45;
}

int main () {
  cequation sqr;
  CRectangle rect;
  sqr.set_side(4);
  rect.convert(sqr);
  cout << rect.area();
  return 0;
}
```