CS304 Midterm Short Notes

Provide by VUAnswer.com

****QUESTION: What is OOP?**
**ANSWER: The object-oriented programming is commonly known as OOP. Most of the languages are developed using OOP concept. Object-oriented programming (OOP) is a programming concept that uses "objects" to develop a system. An object hides the implementation details and exposes only the functionalities and parameters it requires to its client. Here also an object shares the same concept as that of a bike. While driving a motor bike, we are unaware of its implementation details such as how it is developed, internal working of gears etc.? We know only the functions or actions it can perform.

****QUESTION: What are the various elements of OOP?**
**ANSWER: Various elements of OOP are: Object Class Method Encapsulation Information Hiding Inheritance Polymorphism

****QUESTION: What are the characteristics of Object Oriented programming language?**
**ANSWER: Some key features of the Object Oriented programming are: Emphasis on data rather than procedure Programs are divided into entities known as objects Data Structures are designed such that they characterize objects Functions that operate on data of an object are tied together in data structures Data is hidden and cannot be accessed by external functions Objects communicate with each other through functions New data and functions can be easily added whenever necessary Follows bottom up design in program
Design

****QUESTION: What are the basic Concepts used in the Object-Oriented Programming language?**
**ANSWER: Object Class Data Abstraction and Encapsulation Polymorphism Inheritance Message passing Dynamic binding

****QUESTION: What Is an Object? (Object-Oriented Technology)**
**ANSWER: There are many definitions of an object, such as found in [Booch 91, p77]: "An object has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class; the terms instance and object are interchangeable". This is a "classical languages"

definition, as defined in [Coplien 92, p280], where "classes play a central role in the object model", since they do not in prototyping/delegation languages. "The term object was first formally applied in the Simula language, and objects typically existed in Simula programs to simulate some aspect of reality" [Booch 91, p77]. Other definitions referenced by Booch include Smith and Tockey: "an object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain." and [Cox 91]: "anything with a crisply defined boundary" (in context, this is "outside the computer domain". A more conventional definition appears on pg 54). Booch goes on to describe these definitions in depth. [Martin 92, p 241] defines: "An "object" is anything to which a concept applies", and "A concept is an idea or notion we share that applies to certain objects in our awareness". [Rumbaugh 91] defines: "We define an object as a concept, abstraction or thing with crisp boundaries and meaning for the problem at hand." [Shlaer 88, p 14] defines: "An object is an abstraction of a set of real-world things such that:

****QUESTION: What Is Object Encapsulation (Or Protection)?**
**ANSWER: [Booch 91, p. 45] defines: "Encapsulation is the process of hiding all of the details of an object that do not contribute to its essential characteristics." [Coad 91, 1.1.2] defines: "Encapsulation (Information Hiding). A principle, used when developing an overall program structure, that each component of a program should encapsulate or hide a single design decision... The interface to each module is defined in such a way as to reveal as little as possible about its inner workings. [Oxford, 1986]" Some languages permit arbitrary access to objects and allow methods to be defined outside of a class as in conventional programming. Simula and Object Pascal provide no protection for objects, meaning instance variables may be accessed wherever visible. CLOS and Ada allow methods to be defined outside of a class, providing functions and procedures. While both CLOS and Ada have packages for encapsulation, CLOS's are optional while Ada's methodology clearly specifies class-like encapsulation (Adts). However most objectoriented languages provide a well-defined interface to their objects thru classes. C++ has a very general encapsulation/protection mechanism with public, private and protected members. Public members (member data and member functions) may be accessed from anywhere. A Stack's Push and Pop methods will be public. Private members are only accessible from within a class. A Stack's representation, such as a list or array, will usually be private. Protected members are accessible from within a class and also from within subclasses (also called derived classes). A Stack's representation could be declared protected allowing subclass access. C++ also allows a class to specify friends (other (sub)classes and functions), that can access all members (its representation). Eiffel 3.0 allows exporting access to specific classes.

****QUESTION: What Is A Class?**
**ANSWER: A class is a general term denoting classification and also has a new meaning in object-oriented methods. Within the OO context, a class is a specification of structure (instance variables), behavior (methods), and inheritance (parents, or recursive structure and behavior) for objects. As pointed out above, classes can also specify access permissions for clients and derived classes, visibility and member lookup resolution. This is a feature-based or intensional definition, emphasizing a class as a descriptor/constructor of objects (as opposed to a collection of objects, as with the more classical extensional view, which may begin the analysis process). Original Aristotlean classification defines a "class" as a generalization of objects: [Booch 91, p93] "a group,
set, or kind marked by common attributes or a common attribute; a group division, distinction, or rating based on quality, degree of competence, or condition".

***QUESTION: What Is A Meta-Class?**
**ANSWER: Meta-Class is a class' class. If a class is an object, then that object must have a class (in classical OO anyway). Compilers provide an easy way to picture Meta- Classes. Classes must be implemented in some way; perhaps with dictionaries for methods, instances, and parents and methods to perform all the work of being a class. This can be declared in a class named "Meta-Class". The Meta-Class can also provide services to application programs, such as returning a set of all methods, instances or parents for review (or even modification). [Booch 91, p 119] provides another example in Smalltalk with timers. In Smalltalk, the situation is more complex

****QUESTION: What Is Inheritance?**
**ANSWER: Inheritance provides a natural classification for kinds of objects and allowsfor the commonality of objects to be explicitly taken advantage of in modeling and constructing object systems. Natural means we use concepts, classification, and generalization to understand and deal with the complexities of the real world. See the example below using computers. Inheritance is a relationship between classes where one class is the parent base/superclass/ancestor/etc.) class of another. Inheritance provides programming by extension (as opposed to programming by reinvention [LaLonde 90]) and can be used as an is-a-kind-of (or is-a) relationship or for differential programming. Inheritance can also double for assignment

****QUESTION: What Is the Difference Between Object-Based and Object-
Oriented?**

**ANSWER: Object-Based Programming usually refers to objects without inheritance [Cardelli 85] and hence without polymorphism, as in '83 Ada and Modula-2. These languages support abstract data types (Adts) and not classes, which provide inheritance and polymorphism. Ada95 and Modula-3; however, support both inheritance and polymorphism and are object-oriented. [Cardelli 85, p481] state "that a language is object-oriented if and only if it satisfies the following requirements: - It supports objects that are data abstractions with an interface of named operations and a hidden local state. - Objects have an associated type. - Types may inherit attributes from supertypes. Objectoriented = data abstractions + object types + type inheritance These definitions are also
found in [Booch 91, Ch2 and Wegner 87]. [Coad 91] provides another model: Object- Oriented = Classes and Objects + Inheritance + Communication with messages

****QUESTION: What is Abstraction?**
**ANSWER: The importance of abstraction is derived from its ability to hide irrelevant details and from the use of names to reference objects. Abstraction is essential in the construction of programs. It places the emphasis on what an object is or does rather than how it is represented or how it works. Thus, it is the primary means of managing complexity in large programs.

****QUESTION: What is a Class Diagram?**
**ANSWER: A class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and
contents using design elements such as classes, packages and objects.

****QUESTION: What is Method Overriding?**
**ANSWER: Method overriding is a language feature that allows a subclass to override a specific implementation of a method that is already provided by one of its super classes. A subclass can give its own definition of methods but need to have the same signature as the method in its super-class. This means that when overriding a method, the subclass's method has to have the same name and parameter list as the super-class's overridden method.

****QUESTION: What is Operator Overloading?**
**ANSWER: The operator overloading is a specific case of polymorphisms in which some or all of operators like +, - or == are treated as polymorphic (multi) functions and as such have different behaviors depending on the types of its arguments.

****QUESTION: What is Method Overloading?**

**ANSWER: The method overloading is the ability to define several methods (in same class) all with the same name but different on the basis of i) number of parameters ii) types of parameters.

QUESTION: What is Polymorphisms?
**ANSWER: Polymorphism is a generic term that means 'many shapes'. More precisely Polymorphism means the ability to request that the same operations be performed by a wide range of different types of things.

QUESTION: What is Inheritance?
**ANSWER: Ability of a new class to be created, from an existing class by extending it, is called inheritance.

QUESTION: What is a base class?
**ANSWER: When inheritance is used to create a new class from another, the new class is called the subclass or derived class, and the class from which it was derived is called the base class.

QUESTION: What is a concrete class?
**ANSWER: A concrete class is one that can be used to directly create, or instantiate objects, unlike an abstract base class which can only be used as a base class for other classes which eventually lead to concrete classes

QUESTION: What are data members?
**ANSWER: Objects are miniature programs, consisting of both code and data. The code consists of a series of member functions. The data items are called data members.

QUESTION: What is a constructor?
**ANSWER: Objects are complete, miniature programs and, like any good programs, have well defined initialization and termination phases. They have special routines (i.e.member functions) to look after this. The initialization routine is called the constructor,and C++ ensures that every object is properly initialized by calling its constructor. The designer of the object can have more than one constructor, a situation called overloading
and then the compiler will select between them depending on exactly what arguments are passed to the constructor function. However, there must always be a default constructor, to be used when no information is supplied.

QUESTION: What is a destructor?
**ANSWER: The termination routine is called the destructor, and C++ will provide a default if none is supplied. If, during the lifetime of the object, it uses heap memory then the designer of the object must provide a destructor function to release such memory to avoid a memory leak.

QUESTION: What is global variable?

**ANSWER: Global variables can be accessed throughout a program. Another way to put this is to say they have global scope.

****QUESTION: What is local variable?**
**ANSWER: Local variables can only be accessed within the function, or more specifically the compound statement in which they are declared. Another way to put this is to say they have local scope.

****QUESTION: What is a null pointer?**
**ANSWER: A null pointer is a pointer that is currently pointing to nothing. Often pointers are set to zero to make them null pointers or tested against zero to see if they are null or not.

****QUESTION: What is a pointer?**
**ANSWER: A pointer is a variable that holds the address of another variable or object.

****QUESTION: What is meant by protected?**
**ANSWER: The protected keyword in the class statement means that the following members of the class are not available to users of the objects of the class but can be used by any subclass that inherits from it, and consequently forms part of its implementation.
1) Explain what type of copy the default assignment operator "=" does when applied to objects. (shallow copy or deep copy)
2) How we can use the concept of overloading to convert a type according to our own requirements? Give one example.
3) Give the name of two cases where you MUST use initialization list as opposed to assignment in constructors.
4) Can we create an array of objects for a class having user defined constructor? Justify your answer.
5) Fill in the blanks below with public, protected or private keyword.
Public members of base class are _____ members of derived class
Protected members of base class are _____members of derived class.
6) What is simple association? Explain it with the help of an example.
7) The members of a class that can be accessed without creating the object of the class is called.
8) The >= operator can be overloaded.
9) Which of the following operators cannot be overloaded?
The relation operator (>)
(>>)
()
(:J

Question No: 21 (Marks: 3)

Explain what type of copy the default assignment operator "=" does when applied to objects. (shallow copy or deep copy)

Question: How we can use the concept of overloading to convert a type according to our own requirements? Give one example.

Question: The members of a class that can be accessed without creating the object of the class is called vukiduniya.com

Question: Differentiate between Simple association and Aggregation?

Question: Friend function minimizes “Encapsulation”. Describe in your own wordings?

Question:program about constant member function. See page # 100

Question: can binary operators overloaded, how can they overload give an example for any one operator.
Give one-line definition of "Object Orientation".
A type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

Fill in the blanks below with public, protected or private keyword.

a. Public members of base class are _____ public _____ members of derived class
Protected members of base class are __ protected or private _____members of derived class.

What do you mean by the term Abstraction? Why we use it.
Abstraction is way to cope with complexity and it is used to simplify things.
Principle of abstraction:
Capture only those details about an object that are relevant to current perspective
Abstraction
Example:
Suppose we want implement abstraction for the following statement, "Ali is a PhD student and teaches BS Students"
Here object Ali has two perspectives one is his student perspective and second is his teacher perspective
Abstraction Advantages,
Abstraction has following major advantages,

1. It helps us understanding and solving a problem using object-oriented approach as it hides extra irrelevant details of objects.
2. Focusing on single perspective of an object provides us freedom to change implementation for other aspects of for an object later.
Similar to Encapsulation Abstraction is also used for achieving information hiding as we show only relevant details to related objects and hide other details.

Derived class can inherit base class features? Explain it with example.
Inheritance is a process by which an object inherits parent Object quality. inheritance gives reusability, the derived classes have all the features of the base class and the programmer can choose to add new features specific to the newly created derived class. The relationship between a parent and child class under private inheritance is not "is a", but "is implemented as a"Example: father and child relation. father properties power Get the child. Consider a class Man derived class of Monkey an object ofMan inherits some of monkeys qualities and overrides some Qualities vukiduniya.com like walking straight with two legs and have Additional functions like speech etc..
The simple example in C++ is having a class that inherits a data member from its parentclass.

```
class A
{ public: integer d;
};

class B : public A
{ public:
};
```

The class B in inherits the data member d from class A. When one class inherits from another, it acquires all of its methods and data. We can then instantiate an object of class B and call into that data member.

```
void func() { B b; b.d = 10; };
```

Part A.
Suppose we have a furniture store with the following types of furniture, Chairs, Tables, Computer Tables, Dining Tables and Beds.
1. You have to model this store using inheritance by describing base class, derived classes.
2. You also have to describe the IS–A relationship between these classes.

Part B.
What is IS-A relationship, show how it is implemented using c++ code (you do not have to give the detailed code simply show in a single line or two how it will be implemented).

In knowledge representation and object-oriented programming and design, is-a is a relationship where one class D is a subclass of another class B (and so B is a superclassof D). In object-oriented programming the is-a relationship arises in the context of inheritance concept. One can say that "apple" may inherit all the properties common to all fruits, such as being a fleshy container for the seed of a plant.
The is-a relationship is contrasted with the has-a relationship, which constitutes the hierarchy. It may also be contrasted with the instance-of relation: see type-token distinction.
When designing a model (e.g., a computer program) of the real-world relationship between an object and its subordinate, a common error is confusing the relations has-a and is-a.

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 inheritances.

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 Key{ L k; L* kptr; int length;public: Key(L); // ...}; Suppose the following declarations appear later:
Key i; Key c; Keym; The compiler would create three objects.

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 vukiduniya.com 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

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: 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 #include #include #include class equation;class equation {int a, b;public:int c () {return (c);}voidconvert (Cequation);};class linear {private:int side;public:void set_side (int a) {side=a;}friendclass equation;};void equation::convert (Cequation) { a = 23; b = 45;}intmain () { cequation sqr; CRectangle rect; sqr.set_side(4); rect.convert(sqr); cout rect.area();return 0;}

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: 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.

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
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
{
};
```

[IMG]file:///F:/Users/rabnol/AppData/Local/Temp/msohtmlclip1/01/clip_image004.gif[/IMG]

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
#include
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 behaviour 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.

```
#include this will be #include
#include
using namespace std;
template
class MyClass{
public:
MyClass(){
cout"This is class1"endl;
}
};
template
class MyClass{
public:
MyClass(){
cout"This is class2"endl;
}
};
int main(int argc, char *argv[])
{
MyClass c1;
```

```
MyClass c2;
system("PAUSE");
return 0;
}
```

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: 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.

Question No: 17 (Marks: 1)
Give one line definition of "Ternary Association".
Ternary Association
Association between three classes
Diamond
May have association class connected at one corner of the diamond
Example: Association between Insurance Contract, Person, Insurance Policy

Question No: 20 (Marks: 3)

What do you mean by the term Abstraction? Why we use it.
Abstraction is way to cope with complexity and it is used to simplify things.
Principle of abstraction:
Capture only those details about an object that are relevant to current perspective
Abstraction
Example:
Suppose we want implement abstraction for the following statement, "Ali is a PhD student and teaches BS Students"
Here object Ali has two perspectives one is his student perspective and second is his teacher perspective

Abstraction Advantages,
Abstraction has following major advantages,

1. It helps us understanding and solving a problem using object oriented approach as it hides extra irrelevant details of objects.
2. Focusing on single perspective of an object provides us freedom to change implementation for other aspects of for an object later.
Similar to Encapsulation Abstraction is also used for achieving information hiding as we show only relevant details to related objects, and hide other details.

Question No: 21 (Marks: 5)
Derived class can inherit base class features? Explain it with example.
Inheritance is a process by which an object inherits parent
Object quality. inheritance gives reusability, The derived classes have all the features of the base class and the programmer can choose to add new features specific to the newly created derived class. The relationship between a parent and child class under private inheritance is not "is a", but "is implemented as a"

Example: father and child relation. father properties power
Get the child.
Consider a class Man derived class of Monkey an object of
Man inherits some of monkeys qualities and overrides some
Qualities like walking straight with two legs and have
Additional functions like speech etc..
The simple example in C++ is having a class that inherits a data member from its parentclass.

```
class A
{ public: integer d;
};

class B : public A
```

{ public:
};

The class B in inherits the data member d from class A. When one class inherits from another, it acquires all of its methods and data. We can then instantiate an object of class B and call into that data member.
void func() { B b; b.d = 10; };