what is encapsulation??
Containing and hiding information about an object, such as internal data structures and code. Encapsulation isolates the internal complexity of an object's operation from the rest of the application. For example, a client component asking for net revenue from a business object need not know the data's origin.

What is inheritance?
Inheritance allows one class to reuse the state and behavior of another class. The derived class inherits the properties and method implementations of the base class and extends it by overriding methods and adding additional properties and methods.

What is Polymorphism??
Polymorphism allows a client to treat different objects in the same way even if they were created from different classes and exhibit different behaviors. You can use implementation inheritance to achieve polymorphism in languages such as C++ and Java. Base class object’s pointer can invoke methods in derived class objects. You can also achieve polymorphism in C++ by function overloading and operator overloading.

What is constructor or ctor?
Constructor creates an object and initializes it. It also creates vtable for virtual functions. It is different from other methods in a class.

What is destructor?
Destructor usually deletes any extra resources allocated by the object.

What is default constructor?
Constructor with no arguments or all the arguments has default values.

What is copy constructor?
Constructor which initializes the it's object member variables (by shallow copying) with another object of the same class. If you don't implement one in your class then compiler implements one for you.
for example:
Boo Obj1(10); // calling Boo constructor
Boo Obj2(Obj1); // calling boo copy constructor
Boo Obj2 = Obj1;// calling boo copy constructor

When are copy constructors called?
Copy constructors are called in following cases:
a) when a function returns an object of that class by value
b) when the object of that class is passed by value as an argument to a function
c) when you construct an object based on another object of the same class

d) When compiler generates a temporary object

What is assignment operator?
Default assignment operator handles assigning one object to another of the same class.
Member to member copy (shallow copy)

What are all the implicit member functions of the class? Or what are all the functions which compiler implements for us if we don't define one.

default ctor


copy ctor

assignment operator

default destructor

address operator

What is conversion constructor?
Constructor with a single argument makes that constructor as conversion ctor and it can be used for type conversion.

for example:

class Boo
{
    public:
    Boo( int i );
};

Boo BooObject = 10; // assigning int 10 Boo object

What is conversion operator??
Class can have a public method for specific data type conversions.

for example:

class Boo
{
    double value;
    public:
    Boo( int i )
    operator double()
    {
        return value;
    }
};

Boo BooObject;

double i = BooObject; // assigning object to variable i of type double. now conversion
operator gets called to assign the value.

What is diff between malloc()/free() and new/delete?
malloc allocates memory for object in heap but doesn't invoke object's constructor to initialize the object.
new allocates memory and also invokes constructor to initialize the object.
malloc() and free() do not support object semantics
Does not construct and destruct objects
string * ptr = (string *)malloc (sizeof(string))
Are not safe
Does not calculate the size of the objects that it construct
Returns a pointer to void
int *p = (int *) malloc(sizeof(int));
int *p = new int;
Are not extensible
new and delete can be overloaded in a class
"delete" first calls the object's termination routine (i.e. its destructor) and then releases the space the object occupied on the heap memory. If an array of objects was created using new, then delete must be told that it is dealing with an array by preceding the name with an empty []:
Int_t *my_ints = new Int_t[10];
...
delete []my_ints;
What is the diff between "new" and "operator new"?

"operator new" works like malloc.
What is difference between template and macro??
There is no way for the compiler to verify that the macro parameters are of compatible types. The macro is expanded without any special type checking.
If macro parameter has a postincremented variable ( like c++ ), the increment is performed two times.
Because macros are expanded by the preprocessor, compiler error messages will refer to the expanded macro, rather than the macro definition itself. Also, the macro will show up in expanded form during debugging.
for example:
Macro:
#define min(i, j) (i < j ? i : j)
template:
template T min (T i, T j)
{
  return i < j ? i : j;
}
What are C++ storage classes?
auto
register
static
extern
auto: the default. Variables are automatically created and initialized when they are defined and are destroyed at the end of the block containing their definition. They are not visible outside that block.

register: a type of auto variable. A suggestion to the compiler to use a CPU register for performance.

static: a variable that is known only in the function that contains its definition but is never destroyed and retains its value between calls to that function. It exists from the time the program begins execution.

extern: a static variable whose definition and placement is determined when all object and library modules are combined (linked) to form the executable code file. It can be visible outside the file where it is defined.

What are storage qualifiers in C++?
They are:

const
volatile
mutable

Const keyword indicates that memory once initialized, should not be altered by a program.
volatile keyword indicates that the value in the memory location can be altered even though nothing in the program code modifies the contents. For example if you have a pointer to hardware location that contains the time, where hardware changes the value of this pointer variable and not the program. The intent of this keyword to improve the optimization ability of the compiler.

mutable keyword indicates that particular member of a structure or class can be altered even if a particular structure variable, class, or class member function is constant.

struct data
{
    char name[80];
    mutable double salary;
}

const data MyStruct = { "Satish Shetty", 1000 }; // initialized by compiler
strcpy (MyStruct.name, "Shilpa Shetty"); // compiler error
MyStruct.salary = 2000 ; // compiler is happy allowed

What is reference ??
Reference is a name that acts as an alias, or alternative name, for a previously defined variable or an object. Prepending variable with "&" symbol makes it as reference. For example:

int a;
int &b = a;

What is passing by reference?
Method of passing arguments to a function which takes parameter of type reference. For example:

void swap( int & x, int & y )
{
    int temp = x;
x = y;
y = x;
}
int a=2, b=3;
swap( a, b );

Basically, inside the function there won't be any copy of the arguments "x" and "y" instead they refer to original variables a and b. so no extra memory needed to pass arguments and it is more efficient.

When do use "const" reference arguments in function?
a) Using const protects you against programming errors that inadvertently alter data.
b) Using const allows function to process both const and non-const actual arguments, while a function without const in the prototype can only accept non constant arguments.
c) Using a const reference allows the function to generate and use a temporary variable appropriately.

When are temporary variables created by C++ compiler?
Provided that function parameter is a "const reference", compiler generates temporary variable in following 2 ways.
a) The actual argument is the correct type, but it isn't Lvalue

double Cuberoot ( const double & num )
{
    num = num * num * num;
    return num;
}
double temp = 2.0;
double value = cuberoot ( 3.0 + temp ); // argument is a expression and not a Lvalue;
b) The actual argument is of the wrong type, but of a type that can be converted to the correct type

long temp = 3L;
double value = cuberoot ( temp); // long to double conversion

What is virtual function?
When derived class overrides the base class method by redefining the same function, then if client wants to access redefined the method from derived class through a pointer from base class object, then you must define this function in base class as virtual function.

class parent
{
    void Show()
    {
        cout << "i'm parent" << endl;
    }
};
class child: public parent
{
    void Show()
    {
        cout << "i'm child" << endl;
    }
};
parent * parent_object_ptr = new child;
parent_object_ptr->Show() // calls parent->show() i
now we goto virtual world...

```cpp
class parent {
    virtual void Show() {
        cout << "i'm parent" << endl;
    }
};

class child: public parent {
    void Show() {
        cout << "i'm child" << endl;
    }
};

parent * parent_object_ptr = new child;
parent_object_ptr->show() // calls child->show()

1) A modifier, also called a modifying function is a member function that changes the value of at least one data member. In other words, an operation that modifies the state of an object. Modifiers are also known as ‘mutators’. Example: The function mod is a modifier in the following code snippet:

```cpp
class test {
    int x, y;
    public:
    test() {
        x = 0; y = 0;
    }
    void mod() {
        x = 10;
        y = 15;
    }
};
```

**What is an accessor?**

An accessor is a class operation that does not modify the state of an object. The accessor functions need to be declared as const operations.

**Differentiate between a template class and class template?**

Template class: A generic definition or a parameterized class not instantiated until the client provides the needed information. It’s jargon for plain templates. Class template: A class template specifies how individual classes can be constructed much like the way a class specifies how individual objects can be constructed. It’s jargon for plain classes.

**When does a name clash occur?**
A name clash occurs when a name is defined in more than one place. For example, two different class libraries could give two different classes the same name. If you try to use many class libraries at the same time, there is a fair chance that you will be unable to compile or link the program because of name clashes.

**Define namespace?**

It is a feature in C++ to minimize name collisions in the global name space. This namespace keyword assigns a distinct name to a library that allows other libraries to use the same identifier names without creating any name collisions. Furthermore, the compiler uses the namespace signature for differentiating the definitions.

**What is the use of ‘using’ declaration?**

A using declaration makes it possible to use a name from a namespace without the scope operator.

**What is an Iterator class?**

A class that is used to traverse through the objects maintained by a container class. There are five categories of iterators: input iterators, output iterators, forward iterators, bidirectional iterators, random access. An iterator is an entity that gives access to the contents of a container object without violating encapsulation constraints. Access to the contents is granted on a one-at-a-time basis in order. The order can be storage order (as in lists and queues) or some arbitrary order (as in array indices) or according to some ordering relation (as in an ordered binary tree). The iterator is a construct, which provides an interface that, when called, yields either the next element in the container, or some value denoting the fact that there are no more elements to examine. Iterators hide the details of access to and update of the elements of a container class.

The simplest and safest iterators are those that permit read-only access to the contents of a container class.

**What is an incomplete type?**

Incomplete types refers to pointers in which there is non availability of the implementation of the referenced location or it points to some location whose value is not available for modification.

```
int *i=0x400 // i points to address 400
*i=0; //set the value of memory location pointed by i.
```

Incomplete types are otherwise called uninitialized pointers.

**What is a dangling pointer?**

A dangling pointer arises when you use the address of an object after its lifetime is over. This may occur in situations like returning addresses of the automatic variables from a function or using the address of the memory block after it is freed. The following code snippet shows this:

```cpp
class Sample
{
public:
```

---

Courtesy Owaiz Ahmed
int *ptr;
Sample(int i)
{
    ptr = new int(i);
}

~Sample()
{
    delete ptr;
}
void PrintVal()
{
    cout << "The value is " << *ptr;
}

void SomeFunc(Sample x)
{
    cout << "Say i am in someFunc " <<
}

int main()
{
    Sample s1 = 10;
    SomeFunc(s1);
    s1.PrintVal();
}

In the above example when PrintVal() function is called it is called by the pointer that has been freed by the destructor in SomeFunc.

Differentiate between the message and method?

**Message:**
* Objects communicate by sending messages to each other.
* A message is sent to invoke a method.

**Method**
* Provides response to a message.
* It is an implementation of an operation.

What is an adaptor class or Wrapper class?

A class that has no functionality of its own. Its member functions hide the use of a third party software component or an object with the non-compatible interface or a non-object-oriented implementation.

**What is a Null object?**
It is an object of some class whose purpose is to indicate that a real object of that class does not exist. One common use for a null object is a return value from a member function that is supposed to return an object with some specified properties but cannot find such an object.

**What is class invariant?**
A class invariant is a condition that defines all valid states for an object. It is a logical condition to ensure the correct working of a class. Class invariants must hold when an object is created, and they must be preserved under all operations of the class. In particular all class invariants are both preconditions and post-conditions for all operations or member functions of the class.

What do you mean by Stack unwinding?
It is a process during exception handling when the destructor is called for all local objects between the place where the exception was thrown and where it is caught.

**Define precondition and post-condition to a member function?**
Precondition: A precondition is a condition that must be true on entry to a member function. A class is used correctly if preconditions are never false. An operation is not responsible for doing anything sensible if its precondition fails to hold. For example, the interface invariants of stack class say nothing about pushing yet another element on a stack that is already full. We say that isful() is a precondition of the push operation. Post-condition: A post-condition is a condition that must be true on exit from a member function if the precondition was valid on entry to that function. A class is implemented correctly if post-conditions are never false. For example, after pushing an element on the stack, we know that isempty() must necessarily hold. This is a post-condition of the push operation.

**What are the conditions that have to be met for a condition to be an invariant of the class?**
* The condition should hold at the end of every constructor.
* The condition should hold at the end of every mutator (non-const) operation.

**What are proxy objects?**
Objects that stand for other objects are called proxy objects or surrogates.

```cpp
template
class Array2D
{
public:

class Array1D
{
public:
T& operator[] (int index);
const T& operator[] (int index)const;
};

Array1D operator[] (int index);
const Array1D operator[] (int index) const;
};
```

**The following then becomes legal:**
Here data[3] yields an Array1D object and the operator [] invocation on that object yields the float in position(3,6) of the original two dimensional array. Clients of the Array2D class need not be aware of the presence of the Array1D class. Objects of this latter class stand for one-dimensional array objects that, conceptually, do not exist for clients of Array2D. Such clients program as if they were using real, live, two-dimensional arrays. Each Array1D object stands for a one-dimensional array that is absent from a conceptual model used by the clients of Array2D. In the above example, Array1D is a proxy class. Its instances stand for one-dimensional arrays that, conceptually, do not exist.

Name some pure object oriented languages?

Smalltalk, Java, Eiffel, Sather.

What is an orthogonal base class?
If two base classes have no overlapping methods or data they are said to be independent of, or orthogonal to each other. Orthogonal in the sense means that two classes operate in different dimensions and do not interfere with each other in any way. The same derived class may inherit such classes with no difficulty.

What is a node class?
A node class is a class that,
* relies on the base class for services and implementation,
* provides a wider interface to the users than its base class,
* relies primarily on virtual functions in its public interface
* depends on all its direct and indirect base class
* can be understood only in the context of the base class
* can be used as base for further derivation
* can be used to create objects.
A node class is a class that has added new services or functionality beyond the services inherited from its base class.

What is a container class? What are the types of container classes?
A container class is a class that is used to hold objects in memory or external storage. A container class acts as a generic holder. A container class has a predefined behavior and a well-known interface. A container class is a supporting class whose purpose is to hide the topology used for maintaining the list of objects in memory. When a container class contains a group of mixed objects, the container is called a heterogeneous container; when the container is holding a group of objects that are all the same, the container is called a homogeneous container.

How do you write a function that can reverse a linked-list?

Answer1:

```cpp
void reverselist(void)
{
    if(head==0)
        return;
    if(head-
return;
if(head -
{
head-
tail-
}
else
{
node* pre = head;
node* cur = head-
node* curnext = cur-
head-
cur-

for(; curnext!=0; )
{
cur-
pre = cur;
cur = curnext;
curnext = curnext-
}
curnext-
}
}

Answer2:

node* reverselist(node* head)
{
if(0==head || 0==head->next)
//if head->next ==0 should return head instead of 0;
return 0;

{node* prev = head;
node* curr = head->next;
node* next = curr->next;

for(; next!=0; )
{
curr->next = prev;
prev = curr;
curr = next;
next = next->next;
}
curr->next = prev;

head->next = 0;
head = curr;
}

return head;
}

What is polymorphism?
Polymorphism is the idea that a base class can be inherited by several classes. A base class pointer can point to its child class and a base class array can store different child class objects.

How do you find out if a linked-list has an end? (i.e. the list is not a cycle)
You can find out by using 2 pointers. One of them goes 2 nodes each time. The second one goes at 1 nodes each time. If there is a cycle, the one that goes 2 nodes each time will eventually meet the one that goes slower. If that is the case, then you will know the linked-list is a cycle.

How can you tell what shell you are running on UNIX system?
You can do the Echo $RANDOM. It will return a undefined variable if you are from the C-Shell, just a return prompt if you are from the Bourne shell, and a 5 digit random numbers if you are from the Korn shell. You could also do a ps -l and look for the shell with the highest PID.

What is Boyce Codd Normal form?
A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies in F' of the form a->b, where a and b is a subset of R, at least one of the following holds:

* a->b is a trivial functional dependency (b is a subset of a)
* a is a superkey for schema R

What is pure virtual function?
A class is made abstract by declaring one or more of its virtual functions to be pure. A pure virtual function is one with an initializer of = 0 in its declaration

Write a Struct Time where integer m, h, s are its members
struct Time
{
    int m;
    int h;
    int s;
};

How do you traverse a Btree in Backward in-order?
Process the node in the right subtree
Process the root
Process the node in the left subtree

What is the two main roles of Operating System?
As a resource manager
As a virtual machine
In the derived class, which data member of the base class are visible?
In the public and protected sections.