What is C++?

Released in 1985, C++ is an object-oriented programming language created by Bjarne Stroustrup. C++ maintains almost all aspects of the C language, while simplifying memory management and adding several features - including a new datatype known as a class (you will learn more about these later) - to allow object-oriented programming. C++ maintains the features of C which allowed for low-level memory access but also gives the programmer new tools to simplify memory management.

C++ used for:

C++ is a powerful general-purpose programming language. It can be used to create small programs or large applications. It can be used to make CGI scripts or console-only DOS programs. C++ allows you to create programs to do almost anything you need to do. The creator of C++, Bjarne Stroustrup, has put together a partial list of applications written in C++.

What is the difference between declaration and definition?

The declaration tells the compiler that at some later point we plan to present the definition of this declaration.
E.g.: void stars () //function declaration
The definition contains the actual implementation.
E.g.: void stars () // declarator
{
for(int j=10; j > =0; j--) //function body
cout << *;
cout << endl; }

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.

What is the difference between realloc() and free()?

The free subroutine frees a block of memory previously allocated by the malloc subroutine. Undefined results occur if the Pointer parameter is not a valid pointer. If the Pointer parameter is a null value, no action will occur. The realloc subroutine changes the size of the block of memory pointed to by the Pointer parameter to the number of bytes specified by the Size parameter and returns a new pointer to the block. The pointer specified by the Pointer parameter must have been created with the malloc, calloc, or realloc subroutines and not been deallocated with the free or realloc subroutines. Undefined results occur if the Pointer parameter is not a valid pointer.

Anything wrong with this code?
T *p = 0;
delete p;

Yes, the program will crash in an attempt to delete a null pointer.

How do you decide which integer type to use?
It depends on our requirement. When we are required an integer to be stored in 1 byte (means less than or equal to 255) we use short int, for 2 bytes we use int, for 8 bytes we use long int.

A char is for 1-byte integers, a short is for 2-byte integers, an int is generally a 2-byte or 4-byte integer (though not necessarily), a long is a 4-byte integer, and a long long is a 8-byte integer.

What is the difference between declaration and definition?

The declaration tells the compiler that at some later point we plan to present the definition of this declaration.
E.g.: void stars () //function declaration
The definition contains the actual implementation.
E.g.: void stars () // declarator
{
for(int j=10; j > =0; j--) //function body
cout << *;
cout << endl; }

What are the advantages of inheritance?

It permits code reusability. Reusability saves time in program development. It encourages the reuse of proven and debugged high-quality software, thus reducing problem after a system becomes functional.

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

```
void reverselist(void)
{
if(head==0)
return;
if(head->next==0)
return;
if(head->next==tail)
{
head->next = 0;
tail->next = head;
```

```
}
else
{
node* pre = head;
node* cur = head->next;
node* curnext = cur->next;
head->next = 0;
cur-> next = head;

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

curnext->next = cur;
}
}
```

How do I initialize a pointer to a function?
This is the way to initialize a pointer to a function

```
void fun(int a)
{

}

void main()
{
void (*fp)(int);
fp=fun;
fp(1);

}
```

How do you link a C++ program to C functions?
By using the extern "C" linkage specification around the C function declarations.

Explain the scope resolution operator.
It permits a program to reference an identifier in the global scope that has been hidden by another identifier with the same name in the local scope.

What are the differences between a C++ struct and C++ class?
The default member and base-class access specifier are different.

How many ways are there to initialize an int with a constant?
Two.
There are two formats for initializers in C++ as shown in the example that follows. The first format uses the traditional C notation. The second format uses constructor notation.
int foo = 123;
int bar (123);

How does throwing and catching exceptions differ from using setjmp and longjmp?
The throw operation calls the destructors for automatic objects instantiated since entry to the try block.

What is a default constructor?
Default constructor WITH arguments class B { public: B (int m = 0) : n (m) {} int n; }; int main(int argc, char *argv[]) { B b; return 0; }

What is a conversion constructor?
A constructor that accepts one argument of a different type.

What is the difference between a copy constructor and an overloaded assignment operator?
A copy constructor constructs a new object by using the content of the argument object. An overloaded assignment operator assigns the contents of an existing object to another existing object of the same class.

When should you use multiple inheritance?
There are three acceptable answers: "Never," "Rarely," and "When the problem domain cannot be accurately modeled any other way."

Explain the ISA and HASA class relationships. How would you implement each in a class design?
A specialized class "is" a specialization of another class and, therefore, has the ISA relationship with the other class. An Employee ISA Person. This relationship is best implemented with inheritance. Employee is derived from Person. A class may have an instance of another class. For example, an employee "has" a salary, therefore the Employee class has the HASA relationship with the Salary class. This relationship is best implemented by embedding an object of the Salary class in the Employee class.

When is a template a better solution than a base class?
When you are designing a generic class to contain or otherwise manage objects of other types, when the format and behavior of those other types are unimportant to their containment or management, and particularly when those other types are unknown (thus, the generosity) to the designer of the container or manager class.

What is a mutable member?
One that can be modified by the class even when the object of the class or the member function doing the modification is const.

What is an explicit constructor?
A conversion constructor declared with the explicit keyword. The compiler does not use an explicit constructor to implement an implied conversion of types. It's purpose is reserved explicitly for construction.

What do you mean by inline function?
The idea behind inline functions is to insert the code of a called function at the point where the function is called. If done carefully, this can improve the application's performance in exchange for increased compile time and possibly (but not always) an increase in the size of the generated binary executables.

Write a program that ask for user input from 5 to 9 then calculate the average

```
#include "iostream.h"
int main() {
int MAX = 4;
int total = 0;
int average;
int numb;
for (int i=0; i<MAX; i++) {
cout << "Please enter your input between 5 and 9: ";
cin >> numb;
while ( numb<5 || numb>9) {
cout << "Invalid input, please re-enter: ";
cin >> numb;
}
total = total + numb;
}
average = total/MAX;
cout << "The average number is: " << average << "\n";
return 0;
}
```

Write a short code using C++ to print out all odd number from 1 to 100 using a for loop

```
for( unsigned int i = 1; i < = 100; i++ )
if( i & 0x00000001 )
cout << i << \",\";
```

What is public, protected, private?
Public, protected and private are three access specifier in C++.
Public data members and member functions are accessible outside the class.
Protected data members and member functions are only available to derived classes.
Private data members and member functions can't be accessed outside the class. However there is an exception can be using friend classes.

Write a function that swaps the values of two integers, using int* as the argument type.

```
void swap(int* a, int*b) {
int t;
t = *a;
*a = *b;
*b = t;
}
```

Tell how to check whether a linked list is circular.
Create two pointers, each set to the start of the list. Update each as follows:

```
while (pointer1) {
pointer1 = pointer1->next;
pointer2 = pointer2->next; if (pointer2) pointer2=pointer2->next;
if (pointer1 == pointer2) {
print (\"circular\n\");
}
}
```

OK, why does this work?
If a list is circular, at some point pointer2 will wrap around and be either at the item just before pointer1, or the item before that. Either way, it's either 1 or 2 jumps until they meet.

What is virtual constructors/destructors?
Answer1
Virtual destructors:
If an object (with a non-virtual destructor) is destroyed explicitly by applying the delete operator to a base-class pointer to the object, the base-class destructor function (matching the pointer type) is called on the object.
There is a simple solution to this problem declare a virtual base-class destructor.
This makes all derived-class destructors virtual even though they don't have the same name as the base-class destructor. Now, if the object in the hierarchy is destroyed explicitly by applying the delete operator to a base-class pointer to a derived-class object, the destructor for the appropriate class is called. Virtual constructor: Constructors cannot be virtual. Declaring a constructor as a virtual function is a syntax error.

Virtual constructor: Constructors cannot be virtual. Declaring a constructor as a virtual function is a syntax error. Does c++ support multilevel and multiple inheritance?
Yes.

What are the advantages of inheritance?
• It permits code reusability.
• Reusability saves time in program development.
• It encourages the reuse of proven and debugged high-quality software, thus reducing problem after a system becomes functional.

What is the difference between declaration and definition?
The declaration tells the compiler that at some later point we plan to present the definition of this declaration.
E.g.: void stars () //function declaration
The definition contains the actual implementation.
E.g.: void stars () // declarator
{
for(int j=10; j>=0; j--) //function body
cout<<"*";
cout<<endl; }

Does c++ support multilevel and multiple inheritance?
Yes.

What is a template?
Templates allow to create generic functions that admit any data type as parameters and return value without having to overload the function with all the possible data types. Until certain point they fulfill the functionality of a macro. Its prototype is any of the two following ones:

template <class indetifier> function_declaration; template <typename indetifier> function_declaration;
The only difference between both prototypes is the use of keyword class or typename, its use is indistinct since both expressions have exactly the same meaning and behave exactly the same way.

Define a constructor - What it is and how it might be called (2 methods).
constructor is a member function of the class, with the name of the function being the same as the class name. It also specifies how the object should be initialized.

Ways of calling constructor:
1) Implicitly: automatically by complier when an object is created.
2) Calling the constructors explicitly is possible, but it makes the code unverifiable.

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

What are 2 ways of exporting a function from a DLL?
1.Taking a reference to the function from the DLL instance.
2. Using the DLL 's Type Library

What is the difference between an object and a class?
Classes and objects are separate but related concepts. Every object belongs to a class and every class contains one or more related objects.
- A Class is static. All of the attributes of a class are fixed before, during, and after the execution of a program. The attributes of a class don't change.
- The class to which an object belongs is also (usually) static. If a particular object belongs to a certain class at the time that it is created then it almost certainly will still belong to that class right up until the time that it is destroyed.
- An Object on the other hand has a limited lifespan. Objects are created and eventually destroyed. Also during that lifetime, the attributes of the object may undergo significant change.

Suppose that data is an array of 1000 integers. Write a single function call that will sort the 100 elements data [222] through data [321].
quicksort ((data + 222), 100);

What is a class?
Class is a user-defined data type in C++. It can be created to solve a particular kind of problem. After creation the user need not know the specifics of the working of a class.

What is friend function?
As the name suggests, the function acts as a friend to a class. As a friend of a class, it can access its private and protected members. A friend function is not a member of the class. But it must be listed in the class definition.

Which recursive sorting technique always makes recursive calls to sort subarrays that are about half size of the original array?
Mergesort always makes recursive calls to sort subarrays that are about half size of the original array, resulting in O(n log n) time.

What is abstraction?
Abstraction is of the process of hiding unwanted details from the user.

What are virtual functions?
A virtual function allows derived classes to replace the implementation provided by the base class. The compiler makes sure the replacement is always called whenever the object in question is actually of the derived class, even if the object is accessed by a base pointer rather than a derived pointer. This allows algorithms in the base class to be replaced in the derived class, even if users don't know about the derived class.

What is the difference between an external iterator and an internal iterator? Describe an advantage of an external iterator.
An internal iterator is implemented with member functions of the class that has items to step through. .An external iterator is implemented as a separate class that can be "attach" to the object that has items to step through. .An external iterator has the advantage that many difference iterators can be active simultaneously on the same object.

What is a scope resolution operator?
A scope resolution operator (::), can be used to define the member functions of a class outside the class.

What do you mean by pure virtual functions?
A pure virtual member function is a member function that the base class forces derived classes to provide. Normally these member functions have no implementation. Pure virtual functions are equated to zero.
class Shape { public: virtual void draw() = 0; };

What is polymorphism? Explain with an example?
"Poly" means "many" and "morph" means "form". Polymorphism is the ability of an object (or reference) to assume (be replaced by) or become many different forms of object.
Example: function overloading, function overriding, virtual functions. Another example can be a plus '+' sign, used for adding two integers or for using it to concatenate two strings.

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 initiallize 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<class T>
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 }; //initlized by complier

strcpy ( MyStruct.name, "Shilpa Shetty"); // compiler error
MyStruct.salaray = 2000 ; // complier 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 = temp;
}

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 Cube(const double & num)
{
  num = num * num * num;
  return num;

}

double temp = 2.0;
double value = cube(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...

```
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()
```

What is pure virtual function? or what is abstract class?

When you define only function prototype in a base class without implementation and do the complete implementation in derived class. This base class is called abstract class and client won't able to instantiate an object using this base class.

You can make a pure virtual function or abstract class this way..

```
class Boo
{
void foo() = 0;
}

Boo MyBoo; // compilation error
```

What is Memory alignment??

The term alignment primarily means the tendency of an address pointer value to be a multiple of some power of two. So a pointer with two byte alignment has a zero in the least significant bit. And a pointer with four byte alignment has a zero in both the two least significant bits. And so on. More alignment means a longer sequence of zero bits in the lowest bits of a pointer.

What problem does the namespace feature solve?

Multiple providers of libraries might use common global identifiers causing a name collision when an application tries to link with two or more such libraries. The namespace feature surrounds a library's external declarations with a unique namespace that eliminates the potential for those collisions.

namespace [identifier] { namespace-body }

A namespace declaration identifies and assigns a name to a declarative region.
The identifier in a namespace declaration must be unique in the declarative region in which it is used. The identifier is the name of the namespace and is used to reference its members.

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. Something like a pointer.

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.

What do you mean by Stack unwinding?

It is a process during exception handling when the destructor is called for all local objects in the stack between the place where the exception was thrown and where it is caught.

Name the operators that cannot be overloaded??

sizeof, ., .*, .->, ::, ?:

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.

What is inline function??

The __inline keyword tells the compiler to substitute the code within the function definition for every instance of a function call. However, substitution occurs only at the compiler's discretion. For example, the compiler does not inline a function if its address is taken or if it is too large to inline.

What is overloading??

With the C++ language, you can overload functions and operators. Overloading is the practice of supplying more than one definition for a given function name in the same scope.

- Any two functions in a set of overloaded functions must have different argument lists.
- Overloading functions with argument lists of the same types, based on return type alone, is an error.

What is Overriding?

To override a method, a subclass of the class that originally declared the method must declare a method with the same name, return type (or a subclass of that return type), and same parameter

list.
The definition of the method overriding is:
· Must have same method name.
· Must have same data type.
· Must have same argument list.
Overriding a method means that replacing a method functionality in child class. To imply overriding functionality we need parent and child classes. In the child class you define the same method signature as one defined in the parent class.

What is "this" pointer?

The this pointer is a pointer accessible only within the member functions of a class, struct, or union type. It points to the object for which the member function is called. Static member functions do not have a this pointer.

When a nonstatic member function is called for an object, the address of the object is passed as a hidden argument to the function. For example, the following function call

myDate.setMonth(3);

can be interpreted this way:

setMonth(&myDate, 3);

The object's address is available from within the member function as the this pointer. It is legal, though unnecessary, to use the this pointer when referring to members of the class.

What happens when you make call "delete this;" ??

The code has two built-in pitfalls. First, if it executes in a member function for an extern, static, or automatic object, the program will probably crash as soon as the delete statement executes. There is no portable way for an object to tell that it was instantiated on the heap, so the class cannot assert that its object is properly instantiated. Second, when an object commits suicide this way, the using program might not know about its demise. As far as the instantiating program is concerned, the object remains in scope and continues to exist even though the object did itself in. Subsequent dereferencing of the pointer can and usually does lead to disaster.

You should never do this. Since compiler does not know whether the object was allocated on the stack or on the heap, "delete this" could cause a disaster.

How virtual functions are implemented C++?

Virtual functions are implemented using a table of function pointers, called the vtable. There is one entry in the table per virtual function in the class. This table is created by the constructor of the class. When a derived class is constructed, its base class is constructed first which creates the

vtable. If the derived class overrides any of the base classes virtual functions, those entries in the vtable are overwritten by the derived class constructor. This is why you should never call virtual functions from a constructor: because the vtable entries for the object may not have been set up by the derived class constructor yet, so you might end up calling base class implementations of those virtual functions

What is name mangling in C++??

The process of encoding the parameter types with the function/method name into a unique name is called name mangling. The inverse process is called demangling.

For example Foo::bar(int, long) const is mangled as `bar__C3Fooil'.
For a constructor, the method name is left out. That is Foo::Foo(int, long) const is mangled as `__C3Fooil'.

What is the difference between a pointer and a reference?

A reference must always refer to some object and, therefore, must always be initialized; pointers do not have such restrictions. A pointer can be reassigned to point to different objects while a reference always refers to an object with which it was initialized.

How are prefix and postfix versions of operator++() differentiated?

The postfix version of operator++() has a dummy parameter of type int. The prefix version does not have dummy parameter.

What is the difference between const char *myPointer and char *const myPointer?

Const char *myPointer is a non constant pointer to constant data; while char *const myPointer is a constant pointer to non constant data.

How can I handle a constructor that fails?

throw an exception. Constructors don't have a return type, so it's not possible to use return codes. The best way to signal constructor failure is therefore to throw an exception.

How can I handle a destructor that fails?

Write a message to a log-file. But do not throw an exception.
The C++ rule is that you must never throw an exception from a destructor that is being called during the "stack unwinding" process of another exception. For example, if someone says throw Foo(), the stack will be unwound so all the stack frames between the throw Foo() and the } catch (Foo e) { will get popped. This is called stack unwinding.
During stack unwinding, all the local objects in all those stack frames are destructed. If one of those destructors throws an exception (say it throws a Bar object), the C++ runtime system is in a

no-win situation: should it ignore the Bar and end up in the } catch (Foo e) { where it was originally headed? Should it ignore the Foo and look for a } catch (Bar e) { handler? There is no good answer -- either choice loses information.
So the C++ language guarantees that it will call terminate() at this point, and terminate() kills the process. Bang you're dead.

What is Virtual Destructor?

Using virtual destructors, you can destroy objects without knowing their type - the correct destructor for the object is invoked using the virtual function mechanism. Note that destructors can also be declared as pure virtual functions for abstract classes.

if someone will derive from your class, and if someone will say "new Derived", where "Derived" is derived from your class, and if someone will say delete p, where the actual object's type is "Derived" but the pointer p's type is your class.

Can you think of a situation where your program would crash without reaching the breakpoint which you set at the beginning of main()?

C++ allows for dynamic initialization of global variables before main() is invoked. It is possible that initialization of global will invoke some function. If this function crashes the crash will occur before main() is entered.

Name two cases where you MUST use initialization list as opposed to assignment in constructors.

Both non-static const data members and reference data members cannot be assigned values; instead, you should use initialization list to initialize them.

Can you overload a function based only on whether a parameter is a value or a reference?

No. Passing by value and by reference looks identical to the caller.

What are the differences between a C++ struct and C++ class?

The default member and base class access specifiers are different.

The C++ struct has all the features of the class. The only differences are that a struct defaults to public member access and public base class inheritance, and a class defaults to the private access specifier and private base class inheritance.

What does extern "C" int func(int *, Foo) accomplish?

It will turn off "name mangling" for func so that one can link to code compiled by a C compiler.

How do you access the static member of a class?

<ClassName>::<StaticMemberName>

What is multiple inheritance(virtual inheritance)? What are its advantages and disadvantages?

Multiple Inheritance is the process whereby a child can be derived from more than one parent class. The advantage of multiple inheritance is that it allows a class to inherit the functionality of more than one base class thus allowing for modeling of complex relationships. The disadvantage of multiple inheritance is that it can lead to a lot of confusion(ambiguity) when two base classes implement a method with the same name.

What are the access privileges in C++? What is the default access level?

The access privileges in C++ are private, public and protected. The default access level assigned to members of a class is private. Private members of a class are accessible only within the class and by friends of the class. Protected members are accessible by the class itself and it's sub-classes. Public members of a class can be accessed by anyone.

What is a nested class? Why can it be useful?

A nested class is a class enclosed within the scope of another class. For example:

```
// Example 1: Nested class
//
class OuterClass
{
  class NestedClass
  {
    // ...
  };
  // ...
};
```

Nested classes are useful for organizing code and controlling access and dependencies. Nested classes obey access rules just like other parts of a class do; so, in Example 1, if NestedClass is public then any code can name it as OuterClass::NestedClass. Often nested classes contain private implementation details, and are therefore made private; in Example 1, if NestedClass is private, then only OuterClass's members and friends can use NestedClass.

When you instantiate as outer class, it won't instantiate inside class.

What is a local class? Why can it be useful?

local class is a class defined within the scope of a function -- any function, whether a member function or a free function. For example:

```
//  Example 2: Local class
//
int f()
{
  class LocalClass
  {
    // ...
  };
  // ...
};
```

Like nested classes, local classes can be a useful tool for managing code dependencies.

Can a copy constructor accept an object of the same class as parameter, instead of reference of the object?

No. It is specified in the definition of the copy constructor itself. It should generate an error if a programmer specifies a copy constructor with a first argument that is an object and not a reference.

C interview questions and answers(PROGRAM)

1. **What will print out?**

```
main()
{
        char *p1="name";
        char *p2;
        p2=(char*)malloc(20);
        memset (p2, 0, 20);
        while(*p2++ = *p1++);
        printf("%sn",p2);

}
```

Answer:empty string.

2. **What will be printed as the result of the operation below:**

```
main()
{
   int x=20,y=35;
   x=y++ + x++;
   y= ++y + ++x;
   printf("%d%dn",x,y);

}
```

Answer : 5794

3. **What will be printed as the result of the operation below:**

```
main()
{
   int x=5;
   printf("%d,%d,%dn",x,x< <2,x>>2);

}
```

Answer: 5,20,1

4. **What will be printed as the result of the operation below:**

```
#define swap(a,b) a=a+b;b=a-b;a=a-b;

void main()
{
   int x=5, y=10;
   swap (x,y);
   printf("%d %dn",x,y);
   swap2(x,y);
   printf("%d %dn",x,y);
}

int swap2(int a, int b)
{
   int temp;
   temp=a;
   b=a;
   a=temp;
   return 0;
```

```
}
```

Answer: 10, 5
10, 5

5. **What will be printed as the result of the operation below:**

```
main()
{
   char *ptr = " Cisco Systems";
   *ptr++; printf("%sn",ptr);
   ptr++;
   printf("%sn",ptr);

}
```

Answer: Cisco Systems
isco systems

6. **What will be printed as the result of the operation below:**

```
main()
{
   char s1[]="Cisco";
   char s2[]= "systems";
   printf("%s",s1);
}
```

Answer: Cisco

7. **What will be printed as the result of the operation below:**

```
main()
{
   char *p1;
   char *p2;

   p1=(char *)malloc(25);
   p2=(char *)malloc(25);

   strcpy(p1,"Cisco");
   strcpy(p2,"systems");
   strcat(p1,p2);

   printf("%s",p1);
```

```
}
```

Answer: Ciscosystems

8. **The following variable is available in file1.c, who can access it?:**

```
9. static int average;
```

Answer: all the functions in the file1.c can access the variable.

10. **WHat will be the result of the following code?**

```
#define TRUE 0 // some code

while(TRUE)
{

   // some code

}
```

Answer: This will not go into the loop as TRUE is defined as 0.

11. **What will be printed as the result of the operation below:**

```
int x;
int modifyvalue()
{
   return(x+=10);
}

int changevalue(int x)
{
   return(x+=1);
}

void main()
{
   int x=10;
   x++;
   changevalue(x);
   x++;
   modifyvalue();
   printf("First output:%dn",x);
```

```
    x++;
    changevalue(x);
    printf("Second output:%dn",x);
    modifyvalue();
    printf("Third output:%dn",x);

}
```

Answer: 12 , 13 , 13

12. **What will be printed as the result of the operation below:**

```
main()
{
    int x=10, y=15;
    x = x++;
    y = ++y;
    printf("%d %dn",x,y);

}
```

Answer: 11, 16

13. **What will be printed as the result of the operation below:**

```
main()
{
    int a=0;
    if(a==0)
        printf("Cisco Systemsn");
        printf("Cisco Systemsn");

}
```

Answer: Two lines with "Cisco Systems" will be printed.

References…..

http://www.google.com/

http://www.techinterview.com/

Book- C ++ By Shwetank K Gupta

NOTES…