**1. 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++.

**2. 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.

**3. 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.

**4. Base class has some virtual method and derived class has a method with the same name. If we initialize the base class pointer with derived object, calling of that virtual method will result in which method being called?**

a. Base method   
b. Derived method  
Ans. B

**5. What is function overloading and operator overloading?**  
Function overloading: C++ enables several functions of the same name to be defined, as long as these functions have different sets of parameters (at least as far as their types are concerned). This capability is called function overloading. When an overloaded function is called, the C++ compiler selects the proper function by examining the number, types and order of the arguments in the call. Function overloading is commonly used to create several functions of the same name that perform similar tasks but on different data types.

Operator overloading allows existing C++ operators to be redefined so that they work on objects of user-defined classes. Overloaded operators are syntactic sugar for equivalent function calls. They form a pleasant facade that doesn't add anything fundamental to the language (but they can improve understandability and reduce maintenance costs).

**6. 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.

**7.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;   
}

**8. 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;  
}  
}

**9. 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.

**10. 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;  
}

**11. 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 << \",\";

**12. 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.

**13. 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.

**14. What is virtual constructors/destructors?**  
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 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.

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

**16. 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.

**17. 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;  
 }

**18. What is the difference between an array and a list?**

Array is collection of homogeneous elements. List is collection of heterogeneous elements.  
For Array memory allocated is static and continuous. For List memory allocated is dynamic and Random.  
Array: User need not have to keep in track of next memory allocation.  
List: User has to keep in Track of next location where memory is allocated.  
Array uses direct access of stored members, list uses sequencial access for members.  
/With Array you have direct access to memory position 5  
Object x = a[5]; // x takes directly a reference to 5th element of array  
//With the list you have to cross all previous nodes in order to get the 5th node:  
list mylist;  
list::iterator it;  
for( it = list.begin() ; it != list.end() ; it++ )  
{  
if( i==5)  
{  
x = \*it;  
break;  
}  
i++;  
}

**19. 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.

**20. 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.  
class Point2D{  
int x; int y;  
public Point2D() : x(0) , y(0) {} //default (no argument) constructor  
};  
main(){  
Point2D MyPoint; // Implicit Constructor call. In order to allocate memory on stack, the default constructor is implicitly called.  
Point2D \* pPoint = new Point2D(); // Explicit Constructor call. In order to allocate memory on HEAP we call the default constructor.   
You have two pairs: new() and delete() and another pair : alloc() and free().

**21. Explain differences between eg. new() and malloc()**  
1.) “new and delete” are preprocessors while “malloc() and free()” are functions. [we dont use brackets will calling new or delete].  
2.) no need of allocate the memory while using “new” but in “malloc()” we have to use “sizeof()”.  
3.) “new” will initlize the new memory to 0 but “malloc()” gives random value in the new alloted memory location [better to use calloc()]  
new() allocates continous space for the object instace malloc() allocates distributed space.  
new() is castless, meaning that allocates memory for this specific type,  
malloc(), calloc() allocate space for void \* that is cated to the specific class type pointer.

22. What is the difference between class and structure?  
Structure: Initially (in C) a structure was used to bundle different type of data types together to perform a particular functionality. But C++ extended the structure to contain functions also. The major difference is that all declarations inside a structure are by default public.  
Class: Class is a successor of Structure. By default all the members inside the class are private.

**23. What is RTTI?**  
Runtime type identification (RTTI) lets you find the dynamic type of an object when you have only a pointer or a reference to the base type. RTTI is the official way in standard C++ to discover the type of an object and to convert the type of a pointer or reference (that is, dynamic typing).

**24.What is encapsulation?**  
Packaging an object’s variables within its methods is called encapsulation.

25**.Explain term “Polymorphism”** and give an example using eg. SHAPE object: If I have a base class SHAPE, how would I define DRAW methods for two objects CIRCLE and SQUARE“Polymorphism”: A phenomenon which enables an object to react differently to the same function call. in C++ it is attained by using a keyword virtual  
Example  
public class SHAPE  
{  
public virtual void SHAPE::DRAW()=0;  
}  
Note here the function DRAW() is pure virtual which means the sub classes must implement the DRAW() method and SHAPE cannot be instatiated  
public class CIRCLE::public SHAPE  
{  
public void CIRCLE::DRAW()  
{  
// TODO drawing circle  
}  
}  
public class SQUARE::public SHAPE  
{  
public void SQUARE::DRAW()  
{  
// TODO drawing square  
}  
}

now from the user class the calls would be like globally  
SHAPE \*newShape;  
When user action is to draw  
public void MENU::OnClickDrawCircle(){  
newShape = new CIRCLE();  
}

public void MENU::OnClickDrawCircle(){  
newShape = new SQUARE();  
}

the when user actually draws  
public void CANVAS::OnMouseOperations(){  
newShape->DRAW();  
}  
class SHAPE{  
public virtual Draw() = 0; //abstract class with a pure virtual method  
};  
class CIRCLE{  
public int r;  
public virtual Draw() { this->drawCircle(0,0,r); }  
};

class SQURE  
public int a;  
public virtual Draw() { this->drawRectangular(0,0,a,a); }  
};  
Each object is driven down from SHAPE implementing Draw() function in its own way.

**26. What is an object?**  
Object is a software bundle of variables and related methods. Objects have state and behavior.

**27. 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.

**28. What do you mean by inheritance?**  
Inheritance is the process of creating new classes, called derived classes, from existing classes or base classes. The derived class inherits all the capabilities of the base class, but can add embellishments and refinements of its own.

**29.Describe PRIVATE, PROTECTED and PUBLIC – the differences and give examples.**  
class Point2D{  
int x; int y;  
public int color;  
protected bool pinned;  
public Point2D() : x(0) , y(0) {} //default (no argument) constructor  
};  
Point2D MyPoint;  
You cannot directly access private data members when they are declared (implicitly) private:  
MyPoint.x = 5; // Compiler will issue a compile ERROR  
//Nor yoy can see them:  
int x\_dim = MyPoint.x; // Compiler will issue a compile ERROR

On the other hand, you can assign and read the public data members:  
MyPoint.color = 255; // no problem  
int col = MyPoint.color; // no problem  
With protected data members you can read them but not write them: MyPoint.pinned = true; // Compiler will issue a compile ERROR  
bool isPinned = MyPoint.pinned; // no problem

**30. What is namespace?**  
Namespaces allow us to group a set of global classes, objects and/or functions under a name. To say it somehow, they serve to split the global scope in sub-scopes known as namespaces.  
The form to use namespaces is:  
namespace identifier { namespace-body }  
Where identifier is any valid identifier and namespace-body is the set of classes, objects and functions that are included within the namespace. For example:  
namespace general { int a, b; } In this case, a and b are normal variables integrated within the general namespace. In order to access to these variables from outside the namespace we have to use the scope operator ::. For example, to access the previous variables we would have to put:  
general::a general::b  
The functionality of namespaces is specially useful in case that there is a possibility that a global object or function can have the same name than another one, causing a redefinition error.

**31. What is a COPY CONSTRUCTOR and when is it called?**  
A copy constructor is a method that accepts an object of the same class and copies it’s data members to the object on the left part of assignment:  
class Point2D{  
int x; int y;  
public int color;  
protected bool pinned;  
public Point2D() : x(0) , y(0) {} //default (no argument) constructor  
public Point2D( const Point2D & ) ;  
};  
Point2D::Point2D( const Point2D & p )  
{  
this->x = p.x;  
this->y = p.y;  
this->color = p.color;  
this->pinned = p.pinned;  
}

main(){  
Point2D MyPoint;  
MyPoint.color = 345;  
Point2D AnotherPoint = Point2D( MyPoint ); // now AnotherPoint has color = 345

**32. 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-> , 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

**33. What is virtual class and friend class?**  
Friend classes are used when two or more classes are designed to work together and need access to each other's implementation in ways that the rest of the world shouldn't be allowed to have. In other words, they help keep private things private. For instance, it may be desirable for class DatabaseCursor to have more privilege to the internals of class Database than main() has.

**34. What is the word you will use when defining a function in base class to allow this function to be a polimorphic function?**

virtual

**35. What do you mean by binding of data and functions?**

Encapsulation.

**36. 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

**37. 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.

**38. 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.

**39. 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.

**40. 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.

**41. What is abstraction?**

Abstraction is of the process of hiding unwanted details from the user.

**42. 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.

**43.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.

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

**45. 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; };

**46. 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.

**47.What’s the output of the following program? Why?**

#include <stdio.h>  
main()  
{  
typedef union  
{  
int a;  
char b[10];  
float c;  
}  
Union;  
Union x,y = {100};  
x.a = 50;  
strcpy(x.b,\"hello\");  
x.c = 21.50;  
printf(\"Union x : %d %s %f \n\",x.a,x.b,x.c );  
printf(\"Union y :%d %s%f \n\",y.a,y.b,y.c);  
}  
Given inputs X, Y, Z and operations | and & (meaning bitwise OR and AND, respectively)  
What is output equal to in  
output = (X & Y) | (X & Z) | (Y & Z)

**48. Why arrays are usually processed with for loop?**

The real power of arrays comes from their facility of using an index variable to traverse the array, accessing each element with the same expression a[i]. All the is needed to make this work is a iterated statement in which the variable i serves as a counter, incrementing from 0 to a.length -1. That is exactly what a loop does.

**49. What is an HTML tag?**

An HTML tag is a syntactical construct in the HTML language that abbreviates specific instructions to be executed when the HTML script is loaded into a Web browser. It is like a method in Java, a function in C++, a procedure in Pascal, or a subroutine in FORTRAN.

**50.Explain which of the following declarations will compile and what will be constant - a pointer or the value pointed at: \* const char \***  
\* char const \*  
\* char \* const

**51. What problems might the following macro bring to the application?**

#define sq(x) x\*x

**52. Anything wrong with this code?**

T \*p = new T[10];  
delete p;  
Everything is correct, Only the first element of the array will be deleted”, The entire array will be deleted, but only the first element destructor will be called.

**53. Anything wrong with this code?**

T \*p = 0;  
delete p;  
Yes, the program will crash in an attempt to delete a null pointer.

**54. 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.

**55. What does extern mean in a function declaration?**

Using extern in a function declaration we can make a function such that it can used outside the file in which it is defined.  
An extern variable, function definition, or declaration also makes the described variable or function usable by the succeeding part of the current source file. This declaration does not replace the definition. The declaration is used to describe the variable that is externally defined.  
If a declaration for an identifier already exists at file scope, any extern declaration of the same identifier found within a block refers to that same object. If no other declaration for the identifier exists at file scope, the identifier has external linkage.

**56. What can I safely assume about the initial values of variables which are not explicitly initialized?**

It depends on complier which may assign any garbage value to a variable if it is not initialized.

**57. What is the difference between char a[] = “string”; and char \*p = “string”;?**

In the first case 6 bytes are allocated to the variable a which is fixed, where as in the second case if \*p is assigned to some other value the allocate memory can change.

**58. What’s the auto keyword good for?**  
Not much. It declares an object with automatic storage duration. Which means the object will be destroyed at the end of the objects scope. All variables in functions that are not declared as static and not dynamically allocated have automatic storage duration by default.  
For example  
int main()  
{  
int a; //this is the same as writing “auto int a;”  
}  
Local variables occur within a scope; they are “local” to a function. They are often called automatic variables because they automatically come into being when the scope is entered and automatically go away when the scope closes. The keyword auto makes this explicit, but local variables default to auto auto auto auto so it is never necessary to declare something as an auto auto auto auto.

**59. What is the difference between char a[] = “string”; and char \*p = “string”; ?**

a[] = “string”;  
char \*p = “string”;  
The difference is this:  
p is pointing to a constant string, you can never safely say  
p[3]=’x';  
however you can always say a[3]=’x';  
char a[]=”string”; - character array initialization.  
char \*p=”string” ; - non-const pointer to a const-string.( this is permitted only in the case of char pointer in C++ to preserve backward compatibility with C.)

**60. How do I declare an array of N pointers to functions returning pointers to functions returning pointers to characters?**

If you want the code to be even slightly readable, you will use typedefs.  
typedef char\* (\*functiontype\_one)(void);  
typedef functiontype\_one (\*functiontype\_two)(void);  
functiontype\_two myarray[N]; //assuming N is a const integral

**61. What does extern mean in a function declaration?**  
It tells the compiler that a variable or a function exists, even if the compiler hasn’t yet seen it in the file currently being compiled. This variable or function may be defined in another file or further down in the current file.

**62. 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);  
}

**63. How do you link a C++ program to C functions?**

By using the extern "C" linkage specification around the C function declarations.

**64. 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.

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

**66. 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);

**67. 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.

**68. 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; }

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

**70. 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.

**71. 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."

**72. 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.

**73. 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.

**74. 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.

**75. 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.

**76. What is the Standard Template Library (STL)?**

A library of container templates approved by the ANSI committee for inclusion in the standard C++ specification. A programmer who then launches into a discussion of the generic programming model, iterators, allocators, algorithms, and such, has a higher than average understanding of the new technology that STL brings to C++ programming.

**77. Describe run-time type identification.**

The ability to determine at run time the type of an object by using the typeid operator or the dynamic cast operator.

**78. 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.  
This solution assumes that two library vendors don’t use the same namespace identifier, of course.

**79. Are there any new intrinsic (built-in) data types?**

Yes. The ANSI committee added the bool intrinsic type and its true and false value keywords.

80. Will the following program execute?  
void main()  
{  
void \*vptr = (void \*) malloc(sizeof(void));  
vptr++;  
}  
It will throw an error, as arithmetic operations cannot be performed on void pointers.

**81. For the following C program**

#define AREA(x)(3.14\*x\*x)  
main()  
{  
float r1=6.25,r2=2.5,a;  
a=AREA(r1);  
printf("\n Area of the circle is %f", a);  
a=AREA(r2);  
printf("\n Area of the circle is %f", a);  
}  
What is the output?  
Ans. Area of the circle is 122.656250  
Area of the circle is 19.625000

**82. void main()**

{  
int d=5;  
printf("%f",d);  
}  
Ans: Undefined

83. void main()  
{  
int i;  
for(i=1;i<4,i++)  
switch(i)  
case 1: printf("%d",i);break;  
{  
case 2:printf("%d",i);break;  
case 3:printf("%d",i);break;  
}  
switch(i) case 4:printf("%d",i);  
}  
**Ans: 1,2,3,4**

84.void main()  
{  
char \*s="\12345s\n";  
printf("%d",sizeof(s));  
}  
**Ans: 6**

85.void main()  
{  
unsigned i=1; /\* unsigned char k= -1 => k=255; \*/  
signed j=-1; /\* char k= -1 => k=65535 \*/  
/\* unsigned or signed int k= -1 =>k=65535 \*/  
if(i<j)  
printf("less");  
else  
if(i>j)  
printf("greater");  
else  
if(i==j)  
printf("equal");  
}  
Ans: less</j)

86. void main()  
{  
float j;  
j=1000\*1000;  
printf("%f",j);  
}  
1. 1000000  
2. Overflow  
3. Error  
4. None   
**Ans: 4**

**87. How do you declare an array of N pointers to functions returning pointers to functions returning pointers to characters?**

Ans: The first part of this question can be answered in at least three ways:  
1. char \*(\*(\*a[N])())();  
2. Build the declaration up incrementally, using typedefs:  
typedef char \*pc; /\* pointer to char \*/  
typedef pc fpc(); /\* function returning pointer to char \*/  
typedef fpc \*pfpc; /\* pointer to above \*/  
typedef pfpc fpfpc(); /\* function returning... \*/  
typedef fpfpc \*pfpfpc; /\* pointer to... \*/  
pfpfpc a[N]; /\* array of... \*/  
3. Use the cdecl program, which turns English into C and vice versa:  
cdecl> declare a as array of pointer to function returning pointer to function returning pointer to char  
char \*(\*(\*a[])())()

**88.What is a modifier?**

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’.

**89.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

**90.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 class specifies how individual objects can be constructed. It’s jargon for plain classes

**91.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.

**92.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.

**93.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.

**94.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.

**95.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.

**96.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.

**97.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.

**98.What are proxy objects?**

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

**99.Name some pure object oriented languages.**

Smalltalk, Java, Eiffel, Sather.

**100.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.

**101.What is the difference between Mutex and Binary semaphore?**

semaphore is used to synchronize processes. where as mutex is used to provide synchronization between threads running in the same process

**102.What is destructor?**

Destructor usually deletes any extra resources allocated by the object.

**103.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.

**104.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;

**105.What are the defining traits of an object-oriented language?**

The defining traits of an object-oriented langauge are:  
encapsulation  
inheritance  
polymorphism

**106.What is Quadratic Probing?**

The Performance problem encountered by linear probing is caused by the cluster buildup That occurs as a result of the probing sequence. Quadratic probing uses a different sequence to avoid primary clustering.

**107.What is the chaining?**

The Chaining technique basically looks at the hash table as an array of pointers to linked lists. Each slot in the hash table is either empty or simply consists of a pointer to a linked list. You resolve collisions by adding the elements that hash to the same slot to the linked list to which that slot points. At the same time, deletions are easy, You simply delete elements from the linked list.

**108.What is the Hash Function?**

The hash function is an important part of the hashing technique. This function is used to transform the keys into table addresses. The hash function we choose should be easy to compute and should be able to transform the keys into integers in the range 0 to TR-1. Because most of the commonly used hash functions are based on arithmetic operations, We should convert the keys to numbers on which arithmetic operations can be performed

**109.What is an Visualizations?**

The visualization is the basically a way of presentation ,Its just a fancy name for the diagrams, pictures, screen shots, prototypes, and any other visual representations created to help through and design the graphical user interface of your product.

**110.What is virtual inheritance?**

Inheritance is a basically can be private , public, or virtual. With virtual inheritance there is only one copy of each object even if the object appears more than once in the hierarchy.

**111.What is multithreading**

Multithreading is defined as :It is the task of creating a new thread of execution within an existing process rather than starting a new process to begin a function. It is the ability of an operating system to concurrently run programs that have been divided into subcomponents, or threads.

**112.What is the use of using?**

Using is bassically a namespace scope. Its directive used to declare the accessibility of identifiers declared within a namespace scope.

**113.What is the use of exception handling?**

Exception handling is bassically used to detect exceptions becouse it can be taken a corresponding action

**114.What is EOF?**

EOF bassically stands for End of File, It is used to check for the end of file when a file is being read.

**115.Define the parameterized macros?**

Parameterized macros are use for the parameters . It is the one which consist of template with insertion points for the addition of parameters.

**116.What is overflow error?**

Overflow error basically a type of arithmatic errors.It's caused by the result of an arithmetic operation being greater than the actual space provided by the system.

**117.What is a nested class? Why can it be useful?**  
Nested classes basically useful for organizing code and controlling access and dependencies. Nested classes obey access rules just like other parts of a class do.and that class is a class enclosed within the scope of another class.

**118.What are the disadvantages of C++?**

a)It's not pure object oriented programming language.

b)Its a Platform dependent  
c)C++ does not give excellant graphics as compare to java.  
d)Its Not case sensitive.  
e)C++ have Less features as comapred to Java& C#.  
f)Its Not applicable in web enviornment.  
g)Does not provide very strong type-checking.   
h)c++ code is easily prone to errors related to data types, their conversions.  
i)Does not provide efficient means for garbage collection.   
j)No built in support for threads

**119.What is an iterator?**

An iterator is a bassically a type of object that represents a stream of data. It is Unlike a sequence, an iterator can only provide the next item. The for-in statement uses iterators to control the loop, and iterators can also be used in many other contexts

**120.What is the Auto Storage Class?**

Auto Storage Class is bassically 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.

**121.What is callback function?**

Callback function is the type of pointer for a function

**122.What is the use of tellg ()?**

tellg () provides a information about the current position of input/get pointer.

**123.What is the use of tellp ()?**

tellp ()use for the poitner postion :Its provides the current position of output/put pointer

**124.Define the generic programming?**

Generic Programmng is type of method. The method in which generic types are used as arguments in algorithms for different data types and data structures is called generic programming.

**125.What is the use of Microsoft foundation class library?**

The Microsoft Foundation Class Library also called as A Microsoft Foundation Classes or MFC. It is basically a library that wraps portions of the Windows API in C++ classes, and including functionality that enables them to use a default application framework. Classes are defined for many of the handle-managed Windows objects and also for predefined windows and common controls. MFC library would help us reduce the code and development time.