C++ LAB VIVA QUESTIONS

1. What is the difference between interpreters and compilers?

Interpreters read through source code and translate a program, turning the programmer's code, or program instructions, directly into actions. Compilers translate source code into an executable program that can be run at a later time.

2. How do you compile the source code with your compiler?

Every compiler is different. Be sure to check the documentation that came with your compiler.

3. What does the linker do?

The linker's job is to tie together your compiled code with the libraries supplied by your compiler vendor and other sources. The linker lets you build your program in pieces and then link together the pieces into one big program.

4. What are the steps in the development cycle?

Edit source code, compile, link, test, repeat.

5. What is the difference between the compiler and the preprocessor?

Each time you run your compiler, the preprocessor runs first. It reads through your source code and includes the files you've asked for, and performs other housekeeping chores. The preprocessor is discussed in detail on Day 18, "Object-Oriented Analysis and Design."

6. Why is the function main() special?

main() is called automatically, each time your program is executed.

7. What are the two types of comments, and how do they differ?

C++-style comments are two slashes (//), and they comment out any text until the end of the line. C-style comments come in pairs (/* */), and everything between thematching pairs is commented out. You must be careful to ensure you have matchedpairs.

8. Can comments be nested?

Yes, C++-style comments can be nested within C-style comments. You can, in fact,nest C-style comments within C++-style comments, as long as you remember thatthe C++-style comments end at the end of the line.

9. Can comments be longer than one line?

C-style comments can. If you want to extend C++-style comments to a second line, you must put another set of double slashes (//).

10. What are the differences between the function prototype and the function definition?

The function prototype declares the function; the definition defines it. The prototype ends with a semicolon; the definition need not. The declaration can include the keyword inline and default values for the parameters; the definition cannot. The declaration need not include names for the parameters; the definition must.

11. Do the names of parameters have to agree in the prototype, definition, and call to the function?

No. All parameters are identified by position, not name.

12. If a function doesn't return a value, how do you declare the function?

Declare the function to return void.

13. If you don't declare a return value, what type of return value is assumed?

Any function that does not explicitly declare a return type returns int.

14. What is a local variable?

A local variable is a variable passed into or declared within a block, typically a function. It is visible only within the block.

15. What is scope?

Scope refers to the visibility and lifetime of local and global variables. Scope is usually established by a set of braces.

16. What is recursion?

Recursion generally refers to the ability of a function to call itself.

17. When should you use global variables?

Global variables are typically used when many functions need access to the same data. Global variables are very rare in C++; once you know how to create static class variables, you will almost never create global variables.

18.What is function overloading?

Function overloading is the ability to write more than one function with the same name, distinguished by the number or type of the parameters.

19. What is polymorphism?

Polymorphism is the ability to treat many objects of differing but related types

without regard to their differences. In C++, polymorphism is accomplished by using class derivation and virtual functions.

20. Is the declaration of a class its interface or its implementation?
The declaration of a class is its interface; it tells clients of the class how to interact with the class. The implementation of the class is the set of member functions stored--usually in a related CPP file.

21. What is the difference between public and private data members?
Public data members can be accessed by clients of the class. Private data members can be accessed only by member functions of the class.

22. Can member functions be private?
Yes. Both member functions and member data can be private.

23. Can member data be public?
Although member data can be public, it is good programming practice to make it private and to provide public accessor functions to the data.

24. Do class declarations end with a semicolon? Do class method definitions?
Declarations end with a semicolon after the closing brace; function definitions do not.

25. What is the difference between the indirection operator and the address of operator?
The indirection operator returns the value at the address stored in a pointer. The address of operator (&) returns the memory address of the variable.

26. What is the difference between a reference and a pointer?
A reference is an alias, and a pointer is a variable that holds an address. References cannot be null and cannot be assigned to.

27. When must you use a pointer rather than a reference?
When you may need to reassign what is pointed to, or when the pointer may be null.

28. What does new return if there is insufficient memory to make your new object?
A null pointer (0).

29. What is a constant reference?

This is a shorthand way of saying "a reference to a constant object."

30. What is the difference between passing by reference and passing a reference?
Passing by reference means not making a local copy. It can be accomplished by passing a reference or by passing a pointer.

31. When you overload member functions, in what ways must they differ?
Overloaded member functions are functions in a class that share a name but differ in the number or type of their parameters.

32. What is the difference between a declaration and a definition?
A definition sets aside memory, but a declaration does not. Almost all declarations are definitions; the major exceptions are class declarations, function prototypes, and typedef statements.

33. When is the copy constructor called?
Whenever a temporary copy of an object is created. This happens every time an object is passed by value.

34. When is the destructor called?
The destructor is called each time an object is destroyed, either because it goes out of scope or because you call delete on a pointer pointing to it.

35. How does the copy constructor differ from the assignment operator (=)?
The assignment operator acts on an existing object; the copy constructor creates a new one.

36. What is the this pointer?
The this pointer is a hidden parameter in every member function that points to the object itself.

37. How do you differentiate between overloading the prefix and postfix increments?
The prefix operator takes no parameters. The postfix operator takes a single int parameter, which is used as a signal to the compiler that this is the postfix variant.

38. Can you overload the operator+ for short integers?
No, you cannot overload any operator for built-in types.

39. Is it legal in C++ to overload operator++ so that it decrements a value in your class?
It is legal, but it is a bad idea. Operators should be overloaded in a way that is likely

to be readily understood by anyone reading your code.

40. What return value must conversion operators have in their declaration?
None. Like constructors and destructors, they have no return values.

41. What is a v-table?
A v-table, or virtual function table, is a common way for compilers to manage virtual functions in C++. The table keeps a list of the addresses of all the virtual functions and, depending on the runtime type of the object pointed to, invokes the right function.

42. What is a virtual destructor?
A destructor of any class can be declared to be virtual. When the pointer is deleted, the runtime type of the object will be assessed and the correct derived destructor invoked.

43. How do you show the declaration of a virtual constructor?
There are no virtual constructors.

44. How can you create a virtual copy constructor?
By creating a virtual method in your class, which itself calls the copy constructor.

45. How do you invoke a base member function from a derived class in which you've overridden that function?
Base::FunctionName();

46. How do you invoke a base member function from a derived class in which you have not overridden that function?
FunctionName();

47. If a base class declares a function to be virtual, and a derived class does not use the term virtual when overriding that class, is it still virtual when inherited by a third-generation class?
Yes, the virtuality is inherited and cannot be turned off.

48. What is the protected keyword used for?
protected members are accessible to the member functions of derived objects.

49. What is a down cast?
A down cast (also called "casting down") is a declaration that a pointer to a base

class is to be treated as a pointer to a derived class.

50. What is the v-ptr?

The v-ptr, or virtual-function pointer, is an implementation detail of virtual functions. Each object in a class with virtual functions has a v-ptr, which points to the virtual function table for that class.

51. If a round rectangle has straight edges and rounded corners, your RoundRectclass inherits both from Rectangle and from Circle, and they in turn both inheritfrom Shape, how many Shapes are created when you create a RoundRect?

If neither class inherits using the keyword virtual, two Shapes are created: one forRectangle and one for Shape. If the keyword virtual is used for both classes, onlyone shared Shape is created.

52. If Horse and Bird inherit virtual public from Animal, do their constructors initialize the Animal constructor? If Pegasus inherits from both Horse and Bird, how does it initialize Animal's constructor?

Both Horse and Bird initialize their base class, Animal, in their constructors. Pegasus does as well, and when a Pegasus is created, the Horse and Bird initializations of Animal are ignored.

53. Declare a class Vehicle and make it an abstract data type.

```
class Vehicle
{virtual void Move() = 0;
}
```

54. If a base class is an ADT, and it has three pure virtual functions, how many of these functions must be overridden in its derived classes?

None must be overridden unless you want to make the class non-abstract, in which case all three must be overridden.

55. Can static member variables be private?

Yes. They are member variables, and their access can be controlled like any other. If they are private, they can be accessed only by using member functions or, more commonly, static member functions.

56. Show the declaration for a static member variable.

```
static int itsStatic;
```

57. Show the declaration for a static function pointer.

```
static int SomeFunction();
```

58. Show the declaration for a pointer to function returning long and taking an integer parameter.

long (* function)(int);

59. How do you establish an is-a relationship?

With public inheritance.

60. How do you establish a has-a relationship?

With containment; that is, one class has a member that is an object of another type.

61. What is the difference between containment and delegation?

Containment describes the idea of one class having a data member that is an object of another type. Delegation expresses the idea that one class uses another class to accomplish a task or goal. Delegation is usually accomplished by containment.

62. What is the difference between delegation and implemented-in-terms-of?

Delegation expresses the idea that one class uses another class to accomplish a task or goal. Implemented-in-terms-of expresses the idea of inheriting implementation from another class.

63. What is a friend function?

A friend function is a function declared to have access to the protected and private members of your
class.

64. What is a friend class?

A friend class is a class declared so that all its member functions are friend functions of your class.

65. If Dog is a friend of Boy, is Boy a friend of Dog?

No, friendship is not commutative.

66. If Dog is a friend of Boy, and Terrier derives from Dog, is Terrier a friend of Boy?

No, friendship is not inherited.

67. If Dog is a friend of Boy and Boy is a friend of House, is Dog a friend of House?

No, friendship is not associative.

68. Where must the declaration of a friend function appear?

Anywhere within the class declaration. It makes no difference whether you put the declaration within the public:, protected:, or private: access areas.

69. What is the insertion operator and what does it do?
The insertion operator (<<) is a member operator of the ostream object and is usedfor writing to the output device. 70. What is the extraction operator and what doesit do? The extraction operator (>>) is a member operator of the istream object andis used for writing to your program's variables.

71. What are the three forms of cin.get() and what are their differences?
The first form of get() is without parameters. This returns the value of the character found, and will return EOF (end of file) if the end of the file is reached.
The second form of cin.get() takes a character reference as its parameter; that character is filled with the next character in the input stream. The return value is an iostream object.
The third form of cin.get() takes an array, a maximum number of characters to get,and a terminating character. This form of get() fills the array with up to one fewercharacters than the maximum (appending null) unless it reads the terminatingcharacter, in which case it immediately writes a null and leaves the terminatingcharacter in the buffer.

72. What is the difference between cin.read() and cin.getline()?
cin.read() is used for reading binary data structures.
getline() is used to read from the istream's buffer.

73. What is the default width for ouputting a long integer using the insertion operator?
Wide enough to display the entire number.

74. What is the return value of the insertion operator?
A reference to an istream object.

75. What parameter does the constructor to an ofstream object take?
The filename to be opened.

76. What does the ios::ate argument do?
ios::ate places you at the end of the file, but you can write data anywhere in the file.

77. What is an inclusion guard?
Inclusion guards are used to protect a header file from being included into a program more than once.

78. How do you instruct your compiler to print the contents of the intermediate file showing the effects of the preprocessor?
This quiz question must be answered by you, depending on the compiler you are using.

79. What is the difference between #define debug 0 and #undef debug?

#define debug 0 defines the term debug to equal 0 (zero). Everywhere the word debug is found, the character 0 will be substituted. #undef debug removes any definition of debug; when the word debug is found in the file, it will be left unchanged.

80. Name four predefined macros.

__DATE__, __TIME__, __FILE__, __LINE__

81. Why can't you call invariants() as the first line of your constructor?

The job of your constructor is to create the object. The class invariants cannot and should not exist before the object is fully created, so any meaningful use of invariants() will return false until the constructor is finished.

82. What is the difference between object-oriented programming and procedural programming?

Procedural programming focuses on functions separate from data. Objectorientedprogramming
ties data and functionality together into objects, and focuses on
theinteraction among the objects.

83. To what does "event-driven" refer?

Event-driven programs are distinguished by the fact that action is taken only in response to some form of (usually external) simulation, such as a user's keyboard or mouse input.

84. What are the stages in the development cycle?

Typically, the development cycle includes analysis, design, coding, testing, programming, and interaction and feedback among these stages.

85. What is a rooted hierarchy?

A rooted hierarchy is one in which all the classes in the program derive directly or indirectly from a single base class

86. What is a driver program?

A driver program is simply a function that is designed to exercise whatever objects and functions you are currently programming.

87. What is encapsulation?

Encapsulation refers to the (desirable) trait of bringing together in one class all the data and functionality of one discrete entity.

88. What is the difference between a template and a macro?
Templates are built into the C++ language and are type-safe. Macros are implemented by the preprocessor and are not type-safe.

89. What is the difference between the parameter to a template and the parameter to a function?
The parameter to the template creates an instance of the template for each type. If you create six template instances, six different classes or functions are created. The parameters to the function change the behavior or data of the function, but only one function is created.

90. What is the difference between a type-specific template friend class and a general template friend class?
The general template friend function creates one function for every type of the parameterized class; the type-specific function creates a type-specific instance for each instance of the parameterized class.

91. Is it possible to provide special behavior for one instance of a template but not for other instances?
Yes, create a specialized function for the particular instance. In addition to creatingArray::SomeFunction(), also create Array::SomeFunction() to change the behaviorfor integer arrays.

92. How many static variables are created if you put one static member into a template class definition?
One for each instance of the class.

93. What is an exception?
An exception is an object that is created as a result of invoking the keyword throw. It is used to signal an exceptional condition, and is passed up the call stack to the first catch statement that handles its type.

94. What is a try block?
A try block is a set of statements that might generate an exception.

95. What is a catch statement?
A catch statement has a signature of the type of exception it handles. It follows a try block and acts as the receiver of exceptions raised within the try block.

96. What information can an exception contain?
An exception is an object and can contain any information that can be defined within a user-created class.

97. When are exception objects created?

Exception objects are created when you invoke the keyword throw.

98. Should you pass exceptions by value or by reference?

In general, exceptions should be passed by reference. If you don't intend to modify the contents of the exception object, you should pass a const reference.

99. Will a catch statement catch a derived exception if it is looking for the base class?

Yes, if you pass the exception by reference.

100. If there are two catch statements, one for base and one for derived, which should come first?

catch statements are examined in the order they appear in the source code. The first catch statement whose signature matches the exception is used.

101. What does catch(...) mean?

catch(...) will catch any exception of any type.

102. What is a breakpoint?

A breakpoint is a place in the code where the debugger will stop execution.

103 What is the difference between strcpy() and strncpy()?

strcpy(char* destination, char* source) copies source to destination, and puts a null at the end of destination. destination must be large enough to accommodate source, or strcpy() will simply write past the end of the array. strncpy(char* destination char* source, int howmany) will write howmany bytes of source to destination, but will not put a terminating null.

104. What does ctime() do?

ctime() takes a time_t variable and returns an ASCII string with the current time. The time_t variable is typically filled by passing its address to time().

105. What is the function to call to turn an ASCII string into a long?

atol()