1. In C++ 11, you can use smart pointers to dynamically allocate memory and not worry about deleting the memory when you are finished using it.
2. To use any of the smart pointers in C++ 11, you must #include the memory header file with the following directive:

#include <memory>

1. The ampersand (&) is used to dereference a pointer variable in C++.
2. A pointer can be used as a function argument, given the function access to the original argument.
3. C++ does not perform array bounds checking, making it possible for you to assign a pointer the address of an element out of the boundaries of an array.
4. It is legal to subtract a pointer variable from another pointer variable.
5. C++ 11, the nullptr key word was introduced to represent the address 0.
6. An array name is a pointer constant because the address stored in it cannot be changed during runtime.
7. Any arithmetic operation may be performed on pointers.
8. The expression \*s->p; is only meaningful if s is a pointer to a structure and p is a pointer that is a member of that structure.
9. It is possible for a structure to contain as a member a pointer to its own structure type
10. The statement Rectangle \* boxPtr; defines a variable boxPtr to be a pointer pointing to a type Rectangle.
11. With pointer variables you can access, but you cannot modify, data in other variables.
12. An array name is a pointer constant because the address it represents cannot be changed during run-time.
13. Memory cannot be allocated after a program is already running.
14. A pointer with the value 0 (zero) is called the NULL pointer.
15. It is legal to subtract a pointer variable from another pointer variable.
16. The expression s->m has the same meaning as (\*s).m.
17. The expression s->m is meaningful only when s is a pointer to a structure and m is a member of the structure.
18. Assuming myValues is an array of int values, and index is an int variable, both of the following statements do the same thing.

cout << myValues[index] << endl;

cout << \*(myValues + index) << endl;

1. C++ does not perform array bounds checking, making it possible for you to assign a pointer the address of an element out of the boundaries of an array.
2. It is legal to subtract a pointer variable from another pointer variable.
3. With pointer variables you can access, but you cannot modify, data in other variables.
4. Using a binary search, you are more likely to find an item than if you use a linear search.
5. Before you can perform a selection sort, the data must be stored in ascending order.
6. The number of comparisons made by a binary search is expressed in powers of two.
7. The bubble sort is an easy way to arrange data into ascending order, but it cannot arrange data into descending order.
8. If an array is partially initialized, the uninitialized elements will be set to zero.
9. An individual array element can be processed like any other type of C++ variable.
10. If you attempt to store data past an array's boundaries, it is guaranteed that the compiler will issue an error.
11. When you pass an array as an argument to a function, the function will create a local copy of the array.
12. Array names are NOT Pointer constants.
13. \*ptr += 4; adds 4 to the address stored in ptr.
14. vector<int>list2(list1); on execution will create a vector list1 as a copy of already initialized vector list2.
15. push\_back(value); will insert value into the vector and hence right shift all the elements in the vector by 1 unit.
16. Two dimensional arrays can be passed to function, but the row size MUST be specified in the definition of the parameter variable.
17. With pointer variables you can access, but you cannot modify, data in other variables.
18. The number of comparisons made by a binary search in worst case is expressed in powers of two.
19. A vector object automatically expands in size to accomodate the items stored in it.
20. If an integer array is partially initialized, the uninitialized elements will be set to zero.
21. If you attempt to store data past an array's boundaries, it is guaranteed that the compiler will issue an error.
22. When you pass an array as an argument to a function, the function can modify the contents of the array.
23. The amount of memory used by an array depends upon the array's data type and the number of elements in the array.
24. String objects have a member function named c\_str that returns the contents of the object formatted as a null-terminated C-string.
25. When a function is called, flow of control moves to the function's prototype.
26. The open function invoked through an ifstream variable must be passed the name of an existing file.
27. A stack frame is created for all invoked functions except main().
28. An initialization expression may be omitted from the for loop if no initialization is required.
29. A static variable that is defined within a function is initialized only once, the first time the function is called.
30. A local variable and a global variable may not have the same name within the same program.