1. Every C++ program has a function called  **main**
2. The function main returns an **integer**
3. Built in data types:  **char, bool, int, float, double, string**
4. Before an identifier (name) can be used it must be **initialized/declared**
5. The insertion operator << is used for **output**
6. The extraction operator >> is used for **input**
7. **Endl** is the newline character
8. We use #include <**fstream**> for accessing data files
9. Concerning data files, use **open** to prepare a stream for use
10. Concerning data files, use **close** to break the connection between the stream and the variable when you are done with the stream
11. The statement intVar = int (floatVar); is an example of **explicit type coercion**
12. The body of **while loop** may never execute
13. A sentinel-controlled loop is looking for a marker called a **flag** to tell it to stop the loop
14. An **EOF** controlled loop is looking for the operating system End-Of-File marker to indicate the end of the data to be processed
15. The **for** statement is intended to simplify the writing of count controlled loops.
16. When a loop is contained inside another loop, we call this a **nested**  loop
17. The body of a **do-while** will execute at least 1 time
18. When your running program gets a **break** statement, it causes an immediate exit from the innermost switch, while, do-while, or for statement in which it appears.
19. A **function** is a block of code which only runs when it is called
20. You can receive **information** as a result from a function
21. Functions are important for **reusing code.** Define the code once and use it many times
22. There are three components to a C++ function
    1. The **prototype** is a declaration of the identifier used to name the function
    2. The **definition** contains the statements that perform that function’s task
    3. The **call** appears in the client code and is used to invoke a particular function
23. There are two types of C++ functions
    1. **Void** functions do not return any information from the function
    2. **Value returning**  functions do return information from the function
24. The function definition must immediately follow the function prototype. **F**
25. Pass by value passes the **data stored in** the variable location
26. Pass by reference passes the **address** of the variable location
27. A reference variable is a “reference to an existing variable and is created with the **&** operator
28. To pass a variable by reference, we simply declare the **variable** as references rather than as normal variables
29. Concerning a value retuning function definition, an important observation is the **output** matches the data type of the value listed in the return statement
30. A **struct** is a user defined data type. It can store data of multiple data types. Each element is called a member
31. Int values[1000]; This statement declares a **1-Dimensional Array**
32. You cannot pass arrays **by value** in C++
33. Int sample [6][4]; This statement declares a **2-Dimensional Array**
34. A **typedef** provides a level of abstraction away from the actual types being used, allowing you to focus more on the concept of just what a variable should mean
35. Typedef unsigned int score; is an example of using **typedef/user defined variables**

*Linux Review*

1. An **operating system** is a collection of programs that manage the resources of a computer
2. In UNIX and Linux, a folder is known as a **directory**
3. When you first open a terminal window you will be in your **home directory**, the location where your files will be stored
4. Filenames and directory names are **case sensitive**
5. Display current working directory **pwd**
6. List contents of a directory **ls**
7. List contents of a directory in a longer listing  **ls -l**
8. List the contents of a directory including any hidden files **ls -a**
9. Creates new directory within the current directory **mkdir**
10. Change directory **cd**
11. Switch back to parent directory of current working directory **cd –**
12. Copies source file to specified destination file **cp**
13. Removes specified file **rm**
14. Removes specified directory **rmkdir**
15. Display built in manual pages **man**