QUICK REVIEW

1. Functions are like miniature programs and are called modules.
2. Functions enable you to divide a program into manageable tasks.
3. The C++ system provides the standard (predefined) functions.
4. To use a standard function, you must:
    i. Know the name of the header file that contains the function's specification,
    ii. Include that header file in the program, and
    iii. Know the name and type of the function and number and types of the parameters (arguments).
5. There are two types of user-defined functions: value-returning functions and void functions.
6. Variables defined in a function heading are called formal parameters.
7. Expressions, variables, or constant values used in a function call are called actual parameters.
8. In a function call, the number of actual parameters and their types must match with the formal parameters in the order given.
9. To call a function, use its name together with the actual parameter list.
10. A value-returning function returns a value. Therefore, a value-returning function is used (called) in either an expression or an output statement or as a parameter in a function call.
11. The general syntax of a user-defined function is:

    functionType functionName(formal parameter list)

    {

        Statement

    }

12. The line functionType functionName(formal parameter list) is called the function heading (or function header). Statements enclosed between braces ({ and }) are called the body of the function.
13. The function heading and the body of the function are called the definition of the function.
14. If a function has no parameters, you still need the empty parentheses in both the function heading and the function call.
15. A value-returning function returns its value via the return statement.
16. A function can have more than one return statement. However, when- ever a return statement executes in a function, the remaining statements are skipped and the function exits.
17. A return statement returns only one value.
18. A function prototype is the function heading without the body of the function; the function prototype ends with the semicolon.
19. A function prototype announces the function type, as well as the type and number of parameters, used in the function.
20. In a function prototype, the names of the variables in the formal parameter list are optional.
21. Function prototypes help the compiler correctly translate each function call.

22. In a program, function prototypes are placed before every function definition, including the definition of the function main.
23. When you use function prototypes, user-defined functions can appear in any order in the program.
24. When the program executes, the execution always begins with the first statement in the function main.
25. User-defined functions execute only when they are called.
26. A call to a function transfers control from the caller to the called function.
27. In a function call statement, you specify only the actual parameters, not their data type or the function type.
28. When a function exits, the control goes back to the caller.