

Student Name _____

Functions provide a way to encapsulate code so that it can be reused.

Similarly to variables and constants, a function must be declared before it can be implemented.

The declaration of a function is also called a prototype and has the following form:

```
Return_Type function_Name(parameter_declaration_list);
```

The general form of a function definition is as follows:

```
function heading  
function body
```

where the heading has the general form:

```
Return_Type functionName(parameter_declaration_list)  
{  
    body sequence of statements enclosed in curly braces  
}
```

When a function is called, the arguments are associated with the parameters from left to right - first argument with the first parameter, the second argument with the next parameter, and so on, until the matching is complete. There should be the same number of arguments as parameters and each argument's type must be compatible with the type of the corresponding parameter.

Execution transfers from a function back to the caller when a return statement is encountered or the end of the function is reached.

A return statement is not required for functions whose return type is `void`.

As with programs, functions should be thoroughly tested to ensure correctness. This involves writing "driver" programs that call the functions with specific values and display the values they return.

A function's documentation should include a comment that describes its specification.

A specification for a function should include descriptions of values the function receives - parameters and their types; values input to the function; what it returns; values it outputs; preconditions and postconditions.

Local variables, constants and parameters exist only while a function is executing and thus can be accessed only within the function. This means that other functions may reuse the name of a local for some other purpose without causing a conflict.