SECTION 9.1 QUESTIONS

1. Describe an advantage of dividing a program into multiple functions.
2. What is the name of the diagram that shows the functions that make up a program?
3. List two guidelines that you can use to decide what code is a good candidate for being made into a function.
4. Briefly describe top-down and bottom-up design.
5. Write the prototype for the function below.

```
void print_warning()
 {
  cout << "WARNING: Calculations indicate that you will run\n";
  cout << "out of fuel before the next available gas station.\n";
 }
```

6. Write an if structure to be added to the main function below that calls the **print_warning** function above if **fuel_miles < miles_remaining**.

```
int main()
 {
  float miles_remaining, fuel_remaining, mpg, fuel_miles;

  cout << "How many miles are you from the next gas station? ";
  cin >> miles_remaining;
  cout << "How much fuel do you have left (in gallons)? ";
  cin >> fuel_remaining;
  cout << "What is your average miles to the gallon? ";
  cin >> mpg;

  fuel_miles = fuel_remaining * mpg;




  return 0;
 } // end of main function
```

PROBLEM 9.1.1

Write a complete program based on the functions in questions 5 and 6 above. Add another function called **print_okay** that tells the user that he or she will make it to the next gas station. Convert the if structure to an if/else and include the call to the **print_okay** function in the else clause. Save the source code as *GASCHECK.CPP*.

PROBLEM 9.1.2

Write a program that asks the user for an integer. The program should call one of three functions, based on the value entered. If the value is negative, call a function that prints a message indicating that the value is negative. Create similar functions to call when the value is zero and positive. Save the source code as *VALTEST.CPP*.

Data and Functions

When building a program that consists of functions, you must be concerned with how data is made available to the functions. In this section, you will learn about the accessibility of variables in functions and how to get data to and from functions.

SCOPE OF VARIABLES

Note

Local variables are sometimes called **automatic variables** *and global variables are sometimes called* **external variables.**

You have been working mostly with programs that have one function: **main()**. Within **main()**, you declared variables. These variables, however, would be inaccessible outside of **main()**. The "availability" of a variable is known as its *scope*. While this may sound difficult, in C++ the scope of variables is easy to understand.

Variables in C++ can either be local or global. A ***local variable*** is a variable declared within a function and is accessible only within that function. A ***global variable*** is a variable declared before the main function. Global variables are accessible by any function.

Consider the program in Figure 9-5. One variable (**i**) is declared before the main function, making it a global variable. Because **j** and **k** are declared in the main function, they are local to the main function. Therefore, **j** and **k** cannot be used outside of the main function. Within the function named **myfunction**, the variable **l** is declared. It too is local, and accessible only within **myfunction**. After the last statement in **myfunction** is executed, the variable **l** is gone from memory.

If a statement were added to **myfunction** that attempted to access the variable **k**, located in main, an error would result. The variable **k** is accessible only from within the main function. In a similar manner, the variable **l** is inaccessible outside of **myfunction** because it is local to **myfunction**.

EXERCISE 9-3 SCOPE OF VARIABLES

1. Open *SCOPE.CPP*. The program from Figure 9-5 appears on your screen.
2. Compile and run the program as it appears. Study the source code to get clear in your mind where each variable is available.
3. Enter the following statement at the end of **myfunction**.

```
k = i + j;
```

4. Compile the program to see the errors the new statement generates. Your compiler will probably generate an error telling you that the variable **j** and **k** are not defined. The error is generated because **j** and **k** are available only in the main function.
5. Delete the erring statement and close the source code file.

Why have local variables if they are inaccessible to other parts of the program? One reason is that they exist only while the function is executing and