Lab Work 3

Learning Objectives

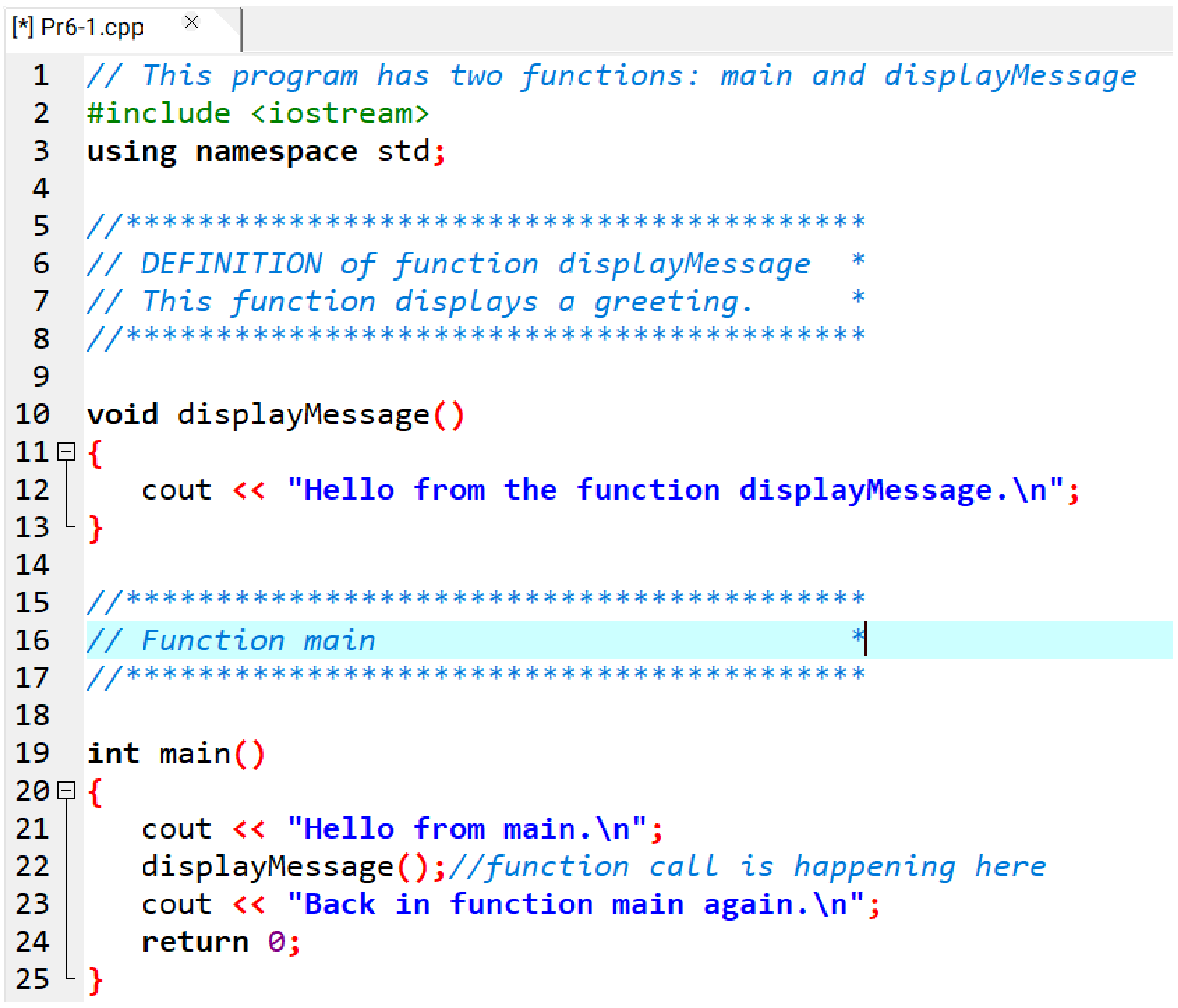
- Introduction to functions
- Defining and calling functions
- Function prototypes
- Sending data through prototypes
  - Passing by value
  - Passing by reference
- The return statement
- Void function
- Local and global variables

Introduction to functions

1. Functions are commonly used to break a problem down into small manageable pieces.
2. Instead of writing one long function that contains all of the statements necessary to solve a problem, several small functions that each solve a specific part of the problem can be written.
3. These small functions can then be executed in the desired order to solve the problem.
4. This approach is sometimes called divide and conquer because a large problem is
5. divided into several smaller problems that are easily solved.

Defining and calling functions

A function call is a statement that causes a function to execute. A function definition contains the statements that make up the function.

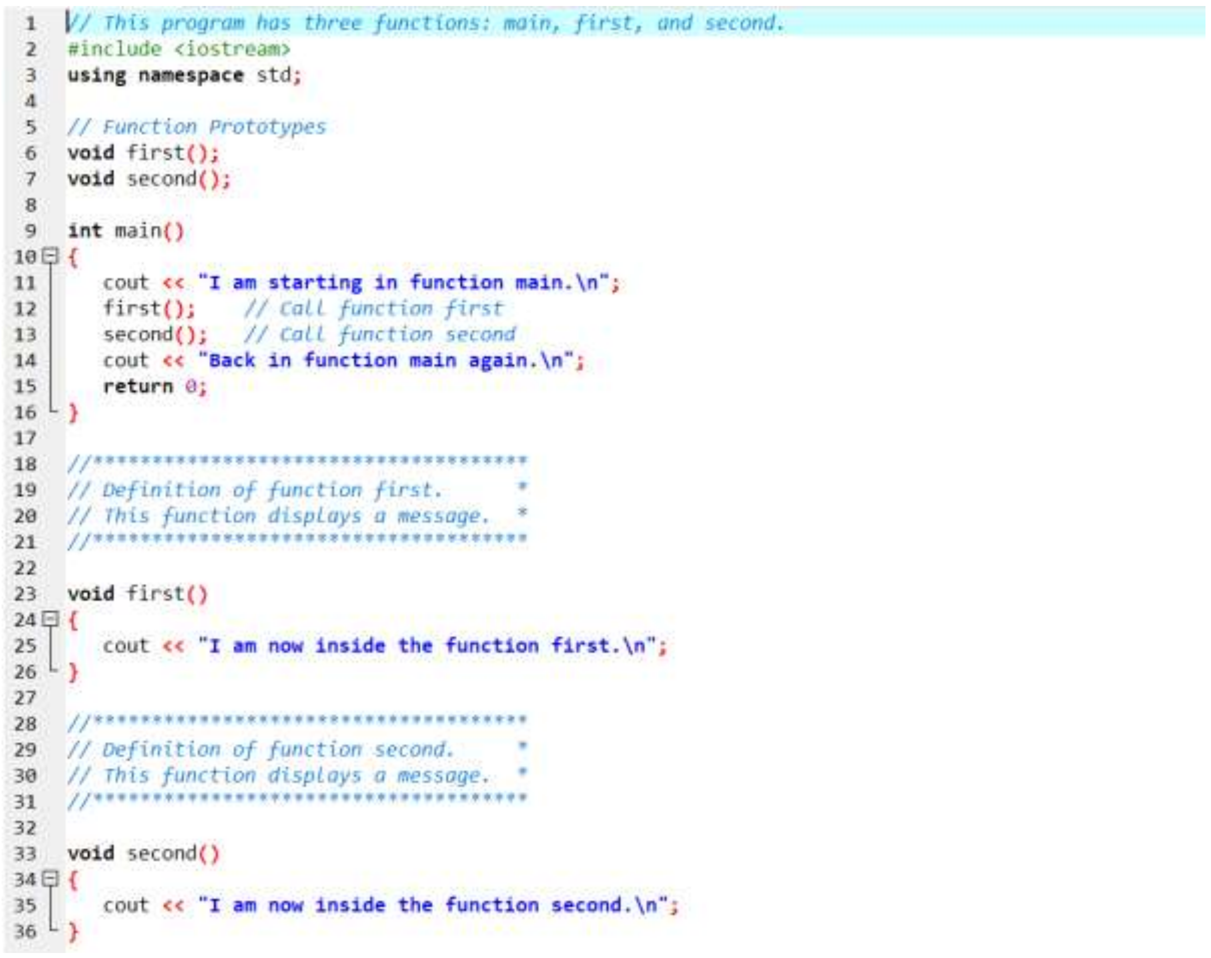


```
1  // This program has two functions: main and displayMessage
2  #include <iostream>
3  using namespace std;
4  
5  //****************************************
6  // DEFINITION of function displayMessage  *
7  // This function displays a greeting.     *
8  //****************************************
9  
10 void displayMessage()
11 {
12     cout << "Hello from the function displayMessage.\n";
13 }
14 
15 //****************************************
16 // Function main                          *
17 //****************************************
18 
19 int main()
20 {
21     cout << "Hello from main.\n";
22     displayMessage();//function call is happening here
23     cout << "Back in function main again.\n";
24     return 0;
25 }
```

Question
Try to indicate which line is showing function definition and which line is showing function call.
Guess what is the output for this program

Function prototypes

A function prototype eliminates the need to place a function definition before all calls to the function.



```
 1  // This program has three functions: main, first, and second.
 2  #include <iostream>
 3  using namespace std;
 4
 5  // Function Prototypes
 6  void first();
 7  void second();
 8
 9  int main()
10  {
11     cout << "I am starting in function main.\n";
12     first();    // Call function first
13     second();   // Call function second
14     cout << "Back in function main again.\n";
15     return 0;
16  }
17
18  //*************************************
19  // Definition of function first.     *
20  // This function displays a message.  *
21  //*************************************
22
23  void first()
24  {
25     cout << "I am now inside the function first.\n";
26  }
27
28  //*************************************
29  // Definition of function second.    *
30  // This function displays a message.  *
31  //*************************************
32
33  void second()
34  {
35     cout << "I am now inside the function second.\n";
36  }
```

Question

Predict what is the output like. Indicate in which line is the function prototype, function declaration and function call.

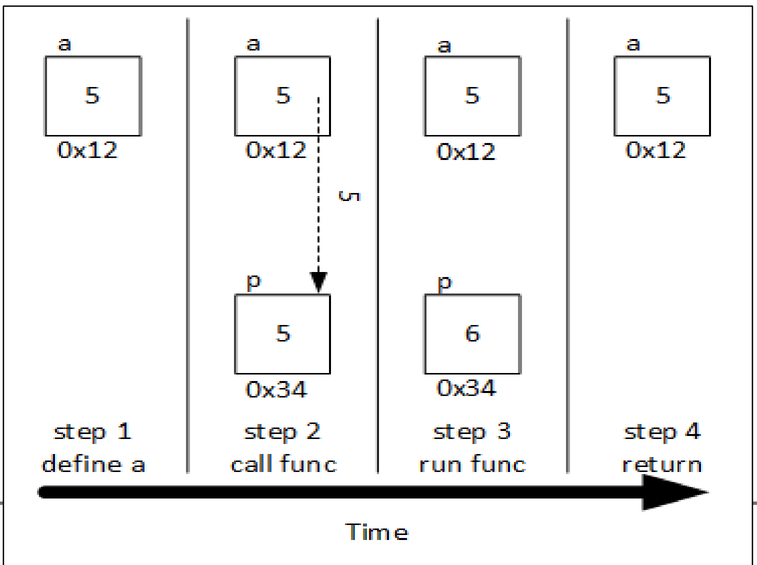
Sending data through prototypes

Passing data by value

When an argument is passed into a parameter, only a copy of the argument's value is passed. Changes to the parameter do not affect the original argument.

Example

```
void func(int p);

int main()
{
        int  a = 5;       // step 1

        func(a);          // step 2
```

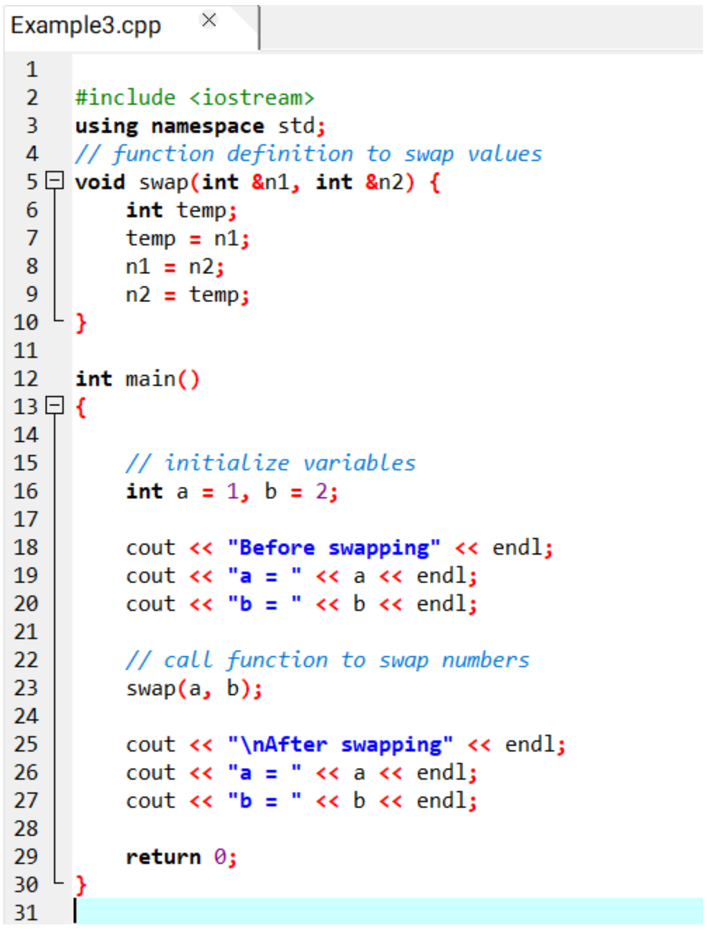


```
}

void func(int p)
{
        p = p + 1;      // step 3
}                       // step 4
```

Note:

1. The variable a is defined and initialized with the value 5
2. The expression in the function call is evaluated - in this example, the expression is very simple and the evaluation is trivial. A copy of the value, 5, is passed from the function call to the local (parameter) variable p (in essence: p = a;)
3. The increment operation takes place in the function; the local variable p is incremented to 6
4. The function terminates and the local variable p is deallocated. Changes to the local variable p do not affect the original argument (variable a) and a remains unchanged

Passing by reference



Example3.cpp

```cpp

#include <iostream>
using namespace std;
// function definition to swap values
void swap(int &n1, int &n2) {
    int temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
}

int main()
{

    // initialize variables
    int a = 1, b = 2;

    cout << "Before swapping" << endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    // call function to swap numbers
    swap(a, b);

    cout << "\nAfter swapping" << endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    return 0;
}

```

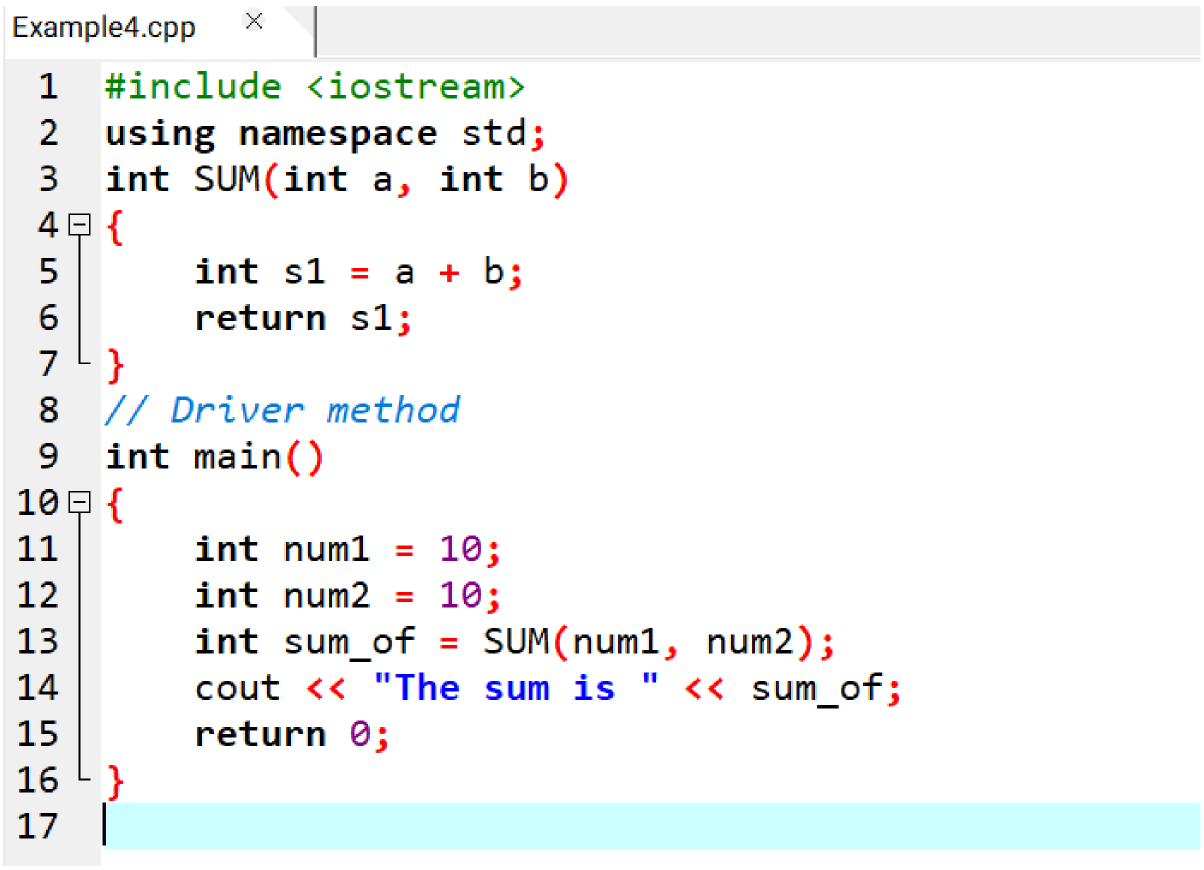
Question

Run the code given above. Write out the output of this code. Draw out a diagram to represent the scenario written in the code above.

The return statements

The return statement causes a function to end immediately.

When the last statement in a void function has finished executing, the function terminates and the program returns to the statement following the function call. It's possible, however, to force a function to return before the last statement has been executed. When the return statement is encountered, the function immediately terminates and control of the program returns to the statement that called the function. The function divide shows the quotient of arg1 divided by arg2 . If arg2 is set to zero, the function returns.



```cpp
#include <iostream>
using namespace std;
int SUM(int a, int b)
{
    int s1 = a + b;
    return s1;
}
// Driver method
int main()
{
    int num1 = 10;
    int num2 = 10;
    int sum_of = SUM(num1, num2);
    cout << "The sum is " << sum_of;
    return 0;
}
```

Question

Run the program and observe the output. Write out your explanation for each line.

Void function

In this case, the type to be used is void, which is a special type to represent the absence of value. For example, a function that simply prints a message may not need to return any value:

```cpp
// void function example
#include <iostream>
using namespace std;
void printmessage ()
{
  cout << "I'm a function!";
}
int main ()
{
  printmessage ();
}
```

Review Questions

1. Write a header for a function named distance. The function should return a double and have two double parameters: rate and time
2. Write a header for a function named days. The function should return an int and have three int parameters: years, months, and weeks.
3. How many return values may a function have?

Local and global variables

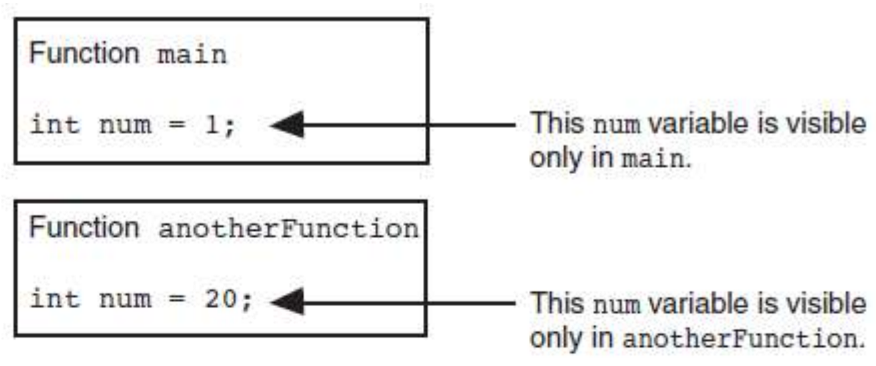
Local Variable

A local variable is defined inside a function and is not accessible outside the function. A global variable is defined outside all functions and is accessible to all functions in its scope.

```
#include <iostream>
using namespace std;

void anotherFunction(); // Function prototype

int main()
{
   int num = 1;    // Local variable

   cout << "In main, num is " << num << endl;
   anotherFunction();
   cout << "Back in main, num is " << num << endl;
   return 0;
}

//*****************************************************
// Definition of anotherFunction                     *
// It has a local variable, num, whose initial value  *
// is displayed.                                      *
//*****************************************************

void anotherFunction()
{
   int num = 20;   // Local variable

   cout << "In anotherFunction, num is " << num << endl;
}
```

Question

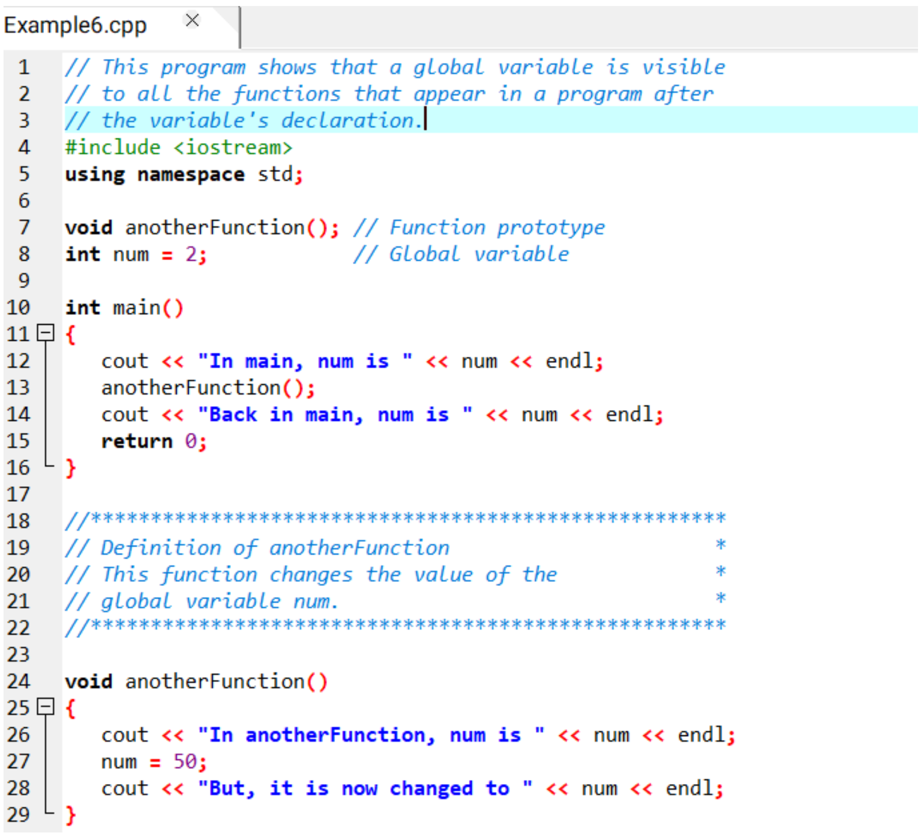
Observe the output for the code above. Try using all the variables declared in the program in both main program and in anotherFunction() function.


Function main
int num = 1;
This num variable is visible only in main.
Function anotherFunction
int num = 20;
This num variable is visible only in anotherFunction.

A function's local variables exist only while the function is executing. This is known as the *lifetime* of a local variable. When the function begins, its local variables and its parameter. variables are created in memory, and when the function ends, the local variables and parameter variables are destroyed. This means that any value stored in a local variable is lost between calls to the function in which the variable is declared.

Global Variable

A global variable is any variable defined outside all the functions in a program. The scope of a global variable is the portion of the program from the variable definition to the end. This means that a global variable can be accessed by all functions that are defined after the
global variable is defined.



Example6.cpp

```
1   // This program shows that a global variable is visible
2   // to all the functions that appear in a program after
3   // the variable's declaration.
4   #include <iostream>
5   using namespace std;
6
7   void anotherFunction(); // Function prototype
8   int num = 2;            // Global variable
9
10  int main()
11  {
12     cout << "In main, num is " << num << endl;
13     anotherFunction();
14     cout << "Back in main, num is " << num << endl;
15     return 0;
16  }
17
18  //*****************************************************
19  // Definition of anotherFunction                      *
20  // This function changes the value of the             *
21  // global variable num.                               *
22  //*****************************************************
23
24  void anotherFunction()
25  {
26     cout << "In anotherFunction, num is " << num << endl;
27     num = 50;
28     cout << "But, it is now changed to " << num << endl;
29  }
```

Question

Run the codes above. Observe the output. Discuss the impact of changes to the program when variable num is used in line 8 and 27.