Activity #11: Void Functions

Recorder's Report

Manager:	Reader:
Recorder:	Driver:
Date:	Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

a) Model 1, Question #3.c

b) Model 2, Question #10

c) Model 3, Question #17

Activity #11: Void Functions

In this activity, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to void functions in C++.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the concept and purpose of a function
- Recognize a function definition, header, and call in a program
- Combine function calls with looping and branching statements
- Develop tests for programs which use functions

Process Skill Goals

During the activity, students should make progress toward:

- Write code that includes function definitions and function calls
- Write programs using functions together with looping and branching statements

Model 1 A C++ Program

```
 1  #include <iostream>
 2
 3  using namespace std;
 4
 5  void printMessage() {
 6    cout << "Welcome to C++." << endl;
 7    cout << "Learn the power of functions!" << endl;
 8  }
 9
10  int main() {
11    cout << "Hello Programmer!" << endl;
12    printMessage();      // function call
13  }
14
```

Output:

```
Hello Programmer!
Welcome to C++.
Learn the power of functions!
```

Refer to Model 1 above as your team develops consensus answers to the questions below.

Questions (10 min) Start time:

1. A *function* is a segment of code that encapsulates a single task. That is, it combines multiple lines of code into a single command.

a) Which lines of code belong to the function `printMessage`?

b) What other function is defined in this code?

c) What does the function `printMessage` do?

2. A *function definition* is the segment of code that tells the program what to do when the function is executed. It contains a *function header* (the first line of the definition) and a *function body* (the commands that make up the function).

a) On what line is the function header for `printMessage`?

b) On what lines is the function body for `printMessage`?

3. A *function call* is a command that executes the function. Use the code found in `activity11a.cpp` to assist you in answering this question.

a) On what line is the function `printMessage` called?

b) What happens if you swap lines 11 and 12 in the model above?

c) What line of code could you add to make the program print the last two lines from the model output twice? Where would you add the code?

Model 2 A C++ Function

```cpp
void printArea(double radius) {
  double area = 3.14159 * pow(radius,2);
  cout << fixed << setprecision(2);
  cout << "The area of a circle with radius "
       << radius << " is " << area << endl;
}

```

Refer to Model 2 above as your team develops consensus answers to the questions below.

Questions (15 min) Start time:

4. What does this function do?

5. Copy down the function header below and underline the name of the function.

6. Other than using a different name, how is this function header different from the header in Model 1?

7. A variable defined in a function header is called a *parameter*. What is the name and purpose of the parameter in this function?

8. Determine the output produced by each of the following calls to this function. You may find it helpful to use the file `activity11b.cpp`.

a) `printArea(3);`

b) `printArea(4.5);`

9. Modify the `main` function in the file `activity11b.cpp` to prompt the user for a radius and then call the function `printArea` with that radius.

10. An *argument* is a value or variable that is passed into the function.

a) What are the arguments in the two function calls in problem 8?

b) What is the argument in your solution to problem 9?

c) What happens to these arguments inside the function?

d) Do variable arguments have to have the same name as the corresponding parameter?

11. Write a function that calculates and prints out the diameter of a circle with a given radius.

12. Add this function to the program in `activity11b.cpp` and add a function call for the same user-entered radius you used in problem #9. Did your function work as expected?

Model 3   A Program with Functions and Branches

```cpp
#include <iostream>

using namespace std;

// first function
void printSum(int num1, int num2) {
  cout << num1 << " + " << num2 << " = " << (num1 + num2) << endl;
}
// second function
void printDifference(int num1, int num2) {
  cout << num1 << " - " << num2 << " = " << (num1 - num2) << endl;
}
// main program
int main() {
  // define variables
  int firstNumber, secondNumber;
  char operation;
  // collect user input
  cout << "Enter a number between 1 and 10: ";
  cin >> firstNumber;
  cout << "Enter another number between 1 and 10: ";
  cin >> secondNumber;
  cout << "Enter a '+' to add or a '-' to subtract: ";
  cin >> operation;
  // decide which function to call
  if (operation == '+') {
    printSum(firstNumber, secondNumber);
  } else if (operation == '-') {
    printDifference(firstNumber, secondNumber);
  } else {
    cout << "Invalid Operation" << endl;
  }
}

```

Refer to Model 3 above as your team develops consensus answers to the questions below.

Questions (25 min)   Start time:

13. What is the first line of code that is executed in this program?

14. Use the file `activity11c.cpp` to execute this program with the following inputs and record the results.

Data Set	First Number	Second Number	Operation	Result
1	2	6	+	
2	3	8	−	
3	34	23	+	
4	4	5	/	

15. What problems do you see when you entered data set 3?

16. Write a function named `checkRange` that takes two integer values as parameters, check to make sure they are both between 1 and 10, and prints out a warning if they are not. For example, calling `checkRange(34,23)` might result in the following output.

```
WARNING! At least one of the numbers you entered was out of range!
```

17. Modify the two functions in the original model to produce the following sample output. Do not change the `main` program. Only change the functions on lines 6-8 and 10-12 of the original model.

Sample Output:

```
Enter a number between 1 and 10:  56
Enter another number between 1 and 10:  4
Enter a '+' to add or a '-' to subtract:  +
56 + 4 = 60
WARNING! At least one of the numbers you entered was out
of range!
```

18. Write a function to draw a frog. Then call this function from the `main` program in a loop to create the output shown.

```
Frog 1...
          @..@
         (----)
        ( >__< )
        ^^ ~~ ^^
Frog 2...
          @..@
         (----)
        ( >__< )
        ^^ ~~ ^^
Frog 3...
          @..@
         (----)
        ( >__< )
        ^^ ~~ ^^
Frog 4...
          @..@
         (----)
        ( >__< )
        ^^ ~~ ^^

```