Module 2: Functions

Learning Objectives

1. Design and implement functions from specifications.
2. Write code that calls functions given function declarations.

Process Skills

1. Information processing. Extract code patterns from sample code.
2. Oral and written communication. Explain code functionality.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. If there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	
Recorder. Records all answers and questions and makes the necessary submission.	

For virtual activities: Once you select your roles, change your Zoom name using the format and example below.

Format: *Group X: First name, Last name initial / Role*

Example: *Group 1: Paul I / Presenter*

Model 1. Value-returning Function (6 min) Start time: _____

Task 1: Create a function that computes the time left in minutes until the donation counter closes at 3:00 pm. The time left is 0 when the current time is after 3:00 pm.

Key Information

Name: Donation time left
Input (parameters): current time in military format (int)
Output (return value/type): time left in minutes (int)
Algorithm (body):

1. Extract hours and minutes from military time.
2. If the current time is after 3:00 pm, give back 0 to the function caller.
3. Subtract current hours from 15 (3:00 in military time).
4. Subtract current minutes from 60. Need to remove the extra hour to account for minutes.
5. Convert hours left to minutes and add the remaining minutes.
6. Give back minutes left to the function caller.

1. Which key information provides a short and clear description of the function's goal?

Name

2. Discuss Task 1 as a group. What information would you need to compute the time left until 3:00 pm? Write your answer below.

Current time

3. How many inputs were required for Task 1? We use the term *parameters* to refer to function inputs.

1

4. Discuss as a group how the *parameters*' data types were identified and select the option that matches your reasoning. For example, using an integer to represent "current time in the military format." Place a check (✓) beside your answer.

    a. We can use an integer to represent any type of input.
    b. Time in the military time format is a number without decimals so we can use an int to represent it. ✓
    c. The data type is incorrect because an int cannot represent AM or PM.

5. Discuss as a group who will most likely use the time left in minutes that was computed by the function. Select the option that matches your reasoning. Place a check (✓) beside your answer.

    a. No one will use the value.
    b. Whoever calls the function can use the value. ✓
    c. The function does not return a value.

6. We use the term *function caller* to refer to whoever uses the function. We use the term *return value* to refer to the data given back by the function. Discuss as a group the relationship between the function caller and the return value. Select the option that matches your reasoning. Place a check (✓) beside your answer.

    a. The function caller accepts the returned value and can use it for other operations. ✓
    b. There is no relationship between the function caller and the return value.
    c. The function caller provides the return value.

7. Discuss as a group how you think the data type of the *return value* was identified. For example, using an integer to represent "time left in minutes." We use the term *return type* to refer to the data type of a function's output. Explain your reasoning by completing the sentence below.

The return type is int because time left in minutes is a numerical value without decimal values.

Model 2. NonValue-returning Function (15 min) Start time: _____

Task 2: Create a function that announces (displays on screen) a donation. The announcement includes the name of the item and its quantity.

Key Information

Name: Announce donation
Input (parameters): name of the food donated (std::string), item quantity (int)
Output (return value/type): None
Algorithm (body):

1. Display the values of the item name and quantity provided.
2. No data is returned to the function caller.

8. Discuss Task 2 as a group. What information would you need to announce a donation? Write your answer below.

Name of food and its quantity.

9. How many *parameters* were required for Task 2?

Two

10. Discuss as a group how the *parameter*'s data types were identified. Explain your reasoning below.

The name of the food is text so it can be represented by a std::string. Quantity is a numerical value without decimals so it can be represented by an int.

We use std::string to represent food name because names are made up of a series of characters.
We use int to represent item quantity because it is numerical and has no decimal values.

11. Work as a group to compare the *return value* of Tasks 1 and 2. Explain why you think we give back information to Task 1's function caller but not Task 2's function caller by completing the sentence below.

Task 2 does not return a value to the function caller because information is already displayed on the screen.

Function design process

1. Read and understand the task/problem.
2. Identify a short name that clearly describes the goal of the function.
3. Identify the parameters you need to perform the given task.
4. Identify the value returned after performing the task.
5. Identify the appropriate data type for the parameters and return value.
6. Design an algorithm that uses the parameters to produce the return value. Go back to step 3 if you need to update your parameters or return value.

12. Apply the function design process to design a function for the task below. Complete the name, input (parameters), output (return value/type), and algorithm sections. We use the term *function body* to refer to the algorithm implemented by the function.

Task 3: Create a function that requests food from the food bank. Requesters will provide the food they want, the quantity they need, and the number of people in their household. The function will tell them the amount of food they can be provided. The food bank only gives enough food for people in a household. For example, a household of 5 people can only get a maximum of 5 sandwiches. If they request 6, they will still get 5.

Key Information

Name: Request Food
Input (parameters): name of food (std::string), quantity (int), household count(int)
Output (return value/type): amount of food (int)
Algorithm (body):

1. Compare the quantity and household count.
2. If quantity is greater than the household count, then return the household count.
3. If quantity is less than or equal to the household count, then return the quantity.

🛑 **STOP HERE AND WAIT FOR FURTHER INSTRUCTIONS**

Model 3. Function definition (10 min) Start time: _____

Task 1

```
// Returns the time in minutes left before donations close for the
// day at 3:00 pm. current_time is expressed as an integer in
// military time. For example: 900, 1330, 1545.
int DonationTimeLeft(int current_time) {
 int current_hr = current_time / 100; // extract first two digits
 int current_min = current_time % 100; // extract last two digits

 if (current_hr >= 15) {
   return 0;
 } else {
   // Get difference between current time and 3:00 pm (15:00).
   // 1 hour is subtracted to account for minutes.
   int hours_left = 15 - current_hr - 1;
   int min_left = 60 - current_min;
   return hours_left * 60 + min_left;
 }
}
```

Task 2

```
// Displays information about a donation on the screen.
void AnnounceDonation(std::string food_name, int quantity) {
 std::cout << quantity << " " << food_name
           << " was donated. Thank you!\n";
}
```

13. Functions begin with a *function prototype*. Map the elements of Task 1's *function prototype* with the information we identified in the function design process. Write *name*, *parameters*, or *return type* in the cells below.

`int`	`DonationTimeLeft`	`(int current_time)`
return type	name	parameters

14. We see two `return` keywords in `DonationTimeLeft`'s *body* (code inside the { }). Work as a team to identify which steps in Task 1's algorithm the `return` keywords perform. Write your answer below.

Steps 2 and 6.

15. What is the relationship between the *return type* and the variable/value written after the `return` keyword? Decide as a group the option that matches your reasoning. Place a check (✓) beside your answer.

    a. The return type is actually based on the parameter's data type.
    b. There is no relationship between the return type and the variable/value.
    c. The return type has the same data type as the variable or value. ✓

16. Write Task 2's function prototype below.

```
void AnnounceDonation(std::string food_name, int quantity)
```

17. Work as a team to analyze Task 2's algorithm. Explain why you think `AnnounceDonation` does not use the `return` keyword by completing the sentence below.

`AnnounceDonation` does not use the `return` keyword because according to Task 2, it does not give back a value to the function caller.

18. Discuss as a group what you think the `void` return type means. Write your answer below. Select the option that matches your reasoning. Place a check (✓) beside your answer.

    a. It is used when the function displays information on the screen.
    b. It is used when the function can return varying types of values. (e.g., returning an int in one case and a std::string in another)
    c. It is used to indicate that the function does not return a value. ✓

Function definition process

1. Create the function prototype using the information you identified during the function design process. Use the name for the function name, inputs for parameters, and outputs for the return type.
2. Use the algorithm from the function design process to write the body of the function.
3. Check to make sure you have a return statement if the function is designed to return data to its user. It should not have a return statement if the function does not return data to its user.

19. We use the term *function definition* to refer to the function prototype and the function body together. Work as a group and use the function definition process to implement the function you designed in Model 2, Task 3. Write the function definition below.

```
int RequestFood(std::string food_name, int quantity,
                int count_household) {
 if (quantity > count_household) {
   return count_household;
 } else {
   return quantity;
 }
}
```

🛑 **STOP HERE AND WAIT FOR FURTHER INSTRUCTIONS**

Model 4. Function calls (12 min) Start time: _____

```
01 int main() {
02  int time_left = DonationTimeLeft(1300);
03  std::cout << "Time left after 1:00 pm: " << time_left
04                                            << " minutes.\n";
05
06  time_left = DonationTimeLeft(1145);
07  std::cout << "Time left after 11:45: " << time_left
08                                          << " minutes. \n";
09
10  AnnounceDonation("Bread", 5);
11  AnnounceDonation("Water", 15);
12  return 0;
13 }
```

20. Which lines of code does your group think calls the `DonationTimeLeft` function? Write the line numbers below.

Lines 2 and 6.

21. What do you think the value `1300` inside the ( ) represents? This is called an *argument*. *Hint: Review the DonationTimeLeft function in Model 2.* Select the option that matches your reasoning. Place a check (✓) beside your answer.

    a. It represents the current time in military format. ✓
    b. It represents the donation cutoff time.
    c. It represents the time left for making donations.

22. Review the `DonationTimeLeft` function as a group. What value do you think the function will return if it is given `1300` as the argument? Write your answer below.

120

23. What information does your group think will be stored in `time_left` after calling `DonationTimeLeft(1300)`?

120

24. Which lines of code does your group think calls the `AnnounceDonation` function? Write the line numbers below.

Lines 10 and 11

25. Discuss as a group why you think `AnnounceDonation` takes two arguments. For example, (`"Bread", 5`). Complete the sentence to express your reasoning.

`AnnounceDonation` takes two arguments because the function prototype requires two parameters.

26. Does calling the `AnnounceDonation` function with different arguments produce the same result? Select the option that matches your reasoning. Place a check (✓) beside your answer.

a. Yes
b. No ✓

27. Work as a group to compare the function call to the `DonationTimeLeft` function and the `AnnounceDonation` function. Why do you think `AnnounceDonation` is not assigned to a variable? Explain your reasoning below.

`AnnounceDonation` is not assigned to a variable because it does not return a value.

Function call process

1. Review the function prototype: name, parameters, and return type.
2. Write the name of the function followed by parentheses.
3. Provide arguments that map to each parameter of the function prototype.
4. If the function returns a value, assign it to a variable with the appropriate type.

28. Work as a group and use the function call process to write a `main` function that calls the `Discard` function shown below. You can use any food as its argument.

```
void Discard(std::string food) {
 std::cout << "Some food got spoiled. Discarding " << food <<
".";
}
```

```
int main() {
 Discard("sandwich");
 return 0;
}
```

29. Work as a group to select one mathematical function from cppreference.com (https://en.cppreference.com/w/cpp/numeric/math). Click on the link to the function and write the *function declaration* below. A *function declaration* is just the function prototype with a semi-colon at the end. Select only one if the website provides more than one function declaration.

```
int abs(int n);
```

30. Work as a group and use the function call process to write a `main` function that calls the cppreference function you selected. Write the code below.

```
int main() {
 int absolute = abs(-4);
 return 0;
}
```

Reflector questions

1. What was the most useful thing your team learned during this session?

2. What was your strategy for finding patterns from the sample code provided?

3. You worked as a group to explain your reasoning about the code you analyzed. Can you think of any benefits in discussing and explaining code?