Homework 4: Arrays and Functions

CS16 - Winter 2021

Due:	Thursday, February 4, 2021 (11:59 PM PST)
Points:	70
Name:	Tyler_Pruitt_____
Homework buddy:	_____

- You may collaborate on this homework with **at most** one person, an optional "homework buddy."
- **Submission instructions:** All questions are to be written (either by hand or typed) *in the provided spaces* and turned in as a single PDF on Gradescope. If you submit handwritten solutions write legibly. We reserve the right to give 0 points to answers we cannot read. When you submit your answer on Gradescope, **be sure to select which portions of your answer correspond to which problem** and clearly mark on the page itself which problem you are answering. We reserve the right to give 0 points to submissions that fail to do this.

1. (2 points) What happens if you forget the return statement in a void function?

Forgetting the return statement in a void function does not produce any errors or warnings and leaving out the return statement is completely acceptable.

2. (2 points) In C++11, can you define a function inside the body of another function? And can you call a function in the body of another function?

In C++11, you cannot define a function inside the body of another function. However, you can call a function inside the body of another function.

3. (2 points) What is the difference between a call-by-reference parameter and a call-by-value parameter? As a programmer, when might you decide to use one over the other?

A call-by-reference parameter is a parameter that when being passed to a

function actually passes the location in memory or memory address of that variable so that if any modifications occur to this call-by-reference parameter, since it is actually the location in memory, the changes will persist to that variable outside of the scope of that function or procedure. A call-by-value parameter is a parameter that when being passed to a function only passes through the value of that parameter when it was passed through so that any modifications to that variable are not in effect outside of the scope of the function because it is only working with the actual value and not the variables memory address. For example, a call-by-value parameter such as 'int num = 5;' will only pass the integer value of 5 to the function and not pass through the memory location of that variable with it. As a programmer, you might want to use call-by-value parameters when you are doing calculations on data that you do not what to change and you might want to use call-by-reference parameters when you are intentionally manipulating data and you want to keep the changes. An example of when to use call-by-value parameters may be when you are computing the determinant of a matrix and you don't want to change the matrix. An example of when to use call-by-reference parameters may be when you are cleaning data, such as cleaning strings.

4. (4 points) Write a void function definition for a function called `zero_both` with 2 parameters, both which are variables of type `int`, and set the value of both variables to 0. Describe if you picked the function parameters to be call-by-reference or call-by-value **and why?**

```
void zero_both(int &num1, int &num2)
{
    //set both variables num1, num2 to zero
    num1 = 0;
    num2 = 0;
}
```

I picked the function parameters to be call-by-reference so that after setting the variables num1,num2 inside of the function to zero the variables num1,num2 will continue to be zero outside of the function scope. Thus, this function definition for zero_both() sets the value of both variables to 0 by using call-by-reference parameters since we want to change the value of the variable at the variable's memory address. I choose call-by-reference because this method enables the function to manipulate the value of the variable with the provided memory address.

5. (8 points) Assume we want to add all the numbers inside of a 3-dimensional int array that is declared as: `int R[3][2][3] = {1};`

a. (2 points) How many loops do we need to use to do this?

We need three (3) loops to add all the numbers inside of a 3-dimensional int array.

b. (2 points) Which would be better to use: `for` loops or `while` loops and **why**?

It would be better to use for loops because we number of iterations of each of the 3 loops is predetermined and we actually know the number of iterations from the int array declaration: 'int R[3][2][3] = {1};'.

c. (2 points) What is the total number of elements in this array?

There are a total of eighteen (18) elements in this array.

d. (2 points) What **values** do the array elements have when the above declaration is executed?

The first element of the array R[0][0][0] is set to one (1), everything else is set to zero. To be more explicit,

R = {{{1, 0, 0}, {0, 0, 0}}, {{0, 0, 0}, {0, 0, 0}}, {{0, 0, 0}, {0, 0, 0}}}

6. (6 points) One way to force a "one time" data type conversion is to use the `static_cast` conversion. Take for example the following code snippet:

```
int number = 7;
int denom = 5;
double var1 = number / denom;
double var2 = static_cast<double>(number) / denom;
int var3 = static_cast<double>(number) / denom;
```

a. (3 points) Would you expect `var1`, `var2`, and `var3` to be equal to each other? Why or why not?

No, I would not expect var1, var2, var3 to be equal to each other because var2 is a double of value 1.4 and var1, var3 are of value 1. This is because var2 is set to a double and it is assigned to the division of a double by an integer so the expression does not evaluate to an integer but a double. Thus, var2 is set to a double of the value 7/5 = 1.4. var3 is an integer set to the integer conversion of 7/5 (rounded down to the nearest integer) so var3 is an integer set to 1. var1 is a double set to the division of two integers, which evaluates to an integer. Thus, since var1 is a double, var1 has the value of 1.00. In conclusion, var1 = 1.00, var2

= 1.4, var3 = 1. Therefore, the variables var1,var2,var3 are not equal to each other.

b. (3 points) Write a function definition for a function that takes one argument of type `int` and one argument of type `double`, and returns a value of type `double` that is the *real number* average of the two arguments.

```
double realAverage(int num1, double num2)
{
    double average;
    average = (num1 + num2) / 2;
    return average;
}
```

7. (30 points) Complete following function definition for the function `find()` that returns *how many times a value appears in an array.* The function takes three arguments: (1) `list`, an array of integers; (2) `asize`, a non-negative integer that indicates the size of the array `list`; (3) `target`, an integer value that is being searched for. All the function should do is return an integer number that indicates how many times the target integer that is being searched for appears in the input array `list`. It should return 0 if that value is not in the array.

```
int find(int list[], int asize, int target)
{
    int count = 0;
    for (int i=0;i<asize;i++)
    {
        if (list[i]==target)
        {
            count++;
        }
    }
    return count;
}
```

8. (6 points) The book and lecture mention variable tracing and stubbing. Describe what they are and how they are best used.

Stubbing is when a particular function is being simplified or streamlined to the point where you are confident that you know the exact behavior of this function because the function you are simplifying is being called by a different function that is being tested. Therefore, a stub is an extremely simplified and easy-to-understand version of a function. Stubbing is best used when you are testing a complete and possibly complicated function and this function definition includes a function call to a separate function, thus it is convenient to simplify the testing of the complicated function by dramatically simplifying the function that is being called so that you know its behavior with certainty and you can better debug and test the function being considered. Stubbing is great for locating and localizing errors or bugs in the code. Variable tracing keeping track of select variables and their values throughout a program to ensure that the variable is being modified (or not modified) in the desired way. Variable tracing is often done by inserting several cout statements the print out the variable name and value throughout the code to let the programmer know what the value of the variable is at location of the cout statement. Variable tracing is best used in loops when a variable may be changing or before, inside, and after functions that use these selected variables are called.

9. (10 points) Write out the contents of a `makefile`, based on the examples from lecture and lab, for a project that compiles a program called `NSA.cpp` that also "includes" a file called `secrets.h`. The compilation must adhere to C++ version 11 standards and should show all warnings. You should also add a `clean` section (again, as per the examples I've shown you). Make sure you use the correct syntax!

```
# NSA.cpp

all: Exercise1

# This compilation command forces the compiler to use the C++ 11 standard.
# -Wall flag shows all warnings (not just errros)
Exercise1: NSA.cpp secrets.h
    g++ NSA.cpp -o NSA -std=c++11 -Wall

clean:
    rm *.0 NSA
```