Assignment #5

CSC111: Introduction to Programming

Learning Outcomes

When you have completed this assignment, you should understand:

- How to define functions that take arrays.
- How to interact with arrays.

Assignment Overview

1. Create a new file called `assignment5.c` in Jupyter Hub.
2. Design the functions according to the specifications in the Function Specifications section.
3. Test to make sure your code produces the expected output.
4. Ensure that the filename is `assignment5.c` (files with an incorrect name will not be marked).
5. Download your `assignment5.c` file from Jupyter Hub (File→Download) and submit this single file to the Assignment 5 dropbox (under Assignments) on the CSC111 BrightSpace Page.
6. Download your file from BrightSpace to confirm that name and file contents are as expected.

Reminder: Your code is to be designed and written by only you and not to be shared with anyone else. See the Course Outline for details explaining the policies on Academic Integrity. Submissions that violate the Academic Integrity policy will be forwarded directly to the Computer Science Academic Integrity Committee.

Grading

1. Late submissions will not be accepted
2. You must submit a single file with the name `assignment5.c` or you will be given a **zero grade**.
3. Your function names must match the specification **exactly** or you will be given a **zero grade**.
4. Any code written in the main function may not be marked.
5. We will do spot-check grading for code quality. All assignments are graded BUT only a subset of your code may be graded for code quality. You will not know which portions of the code will be graded, so all of your code must be complete and adhere to specifications to receive marks.
6. Your code must **compile and run, as submitted,** without errors in the Computer Science department's Jupyter Hub environment using the compilation command provided in this document.
7. We reserve the right to replace your main function with our own, allowing it to better interact with our testing software.
8. It is the responsibility of the student to submit any and all correct files. Only submitted files will be marked. Submitting an incorrect file or forgetting a file is not grounds for a regrade.

Marks will be given for:

- Your code, as submitted, compiling without errors.
- Your code producing the correct output.
- Your code following good coding conventions
  - Proper indentation
  - Documentation (comments)
  - Use of whitespace to improve readability
  - Names of variables should have meaning relevant to what they store
  - Proper use of helper functions

Compiling and Running

Use the following commands to compile and run your code when you are testing your work. Please note the removal of the -Pedantic flag

```
gcc -Wall -Werror -std=c18 -o assignment5 assignment5.c
```

and the following to run the code:

```
./assignment5
```

NOTE: Where examples are provided for clarity in the problem descriptions, they do not necessarily consider all edge cases – your testing must take into account all edge cases as we will when grading your submissions.

Functions Specifications

IMPORTANT: Be careful with your choice of types! All floating point return types and argument types **must** use type **double**. When the return type or argument type is clearly not a fractional number, you **must** use the type **int**.

NOTE: Your function prototypes MUST list the parameters in the order they are described within the function specification!

This section will describe the functions that you are tasked to create.

1. Design a function, called `print_array`, that meets the following criteria:
   - The function must take as arguments an array of integers and the number of elements in the array.
   - The function should print the elements of the array separated by commas. Do not print a comma after the final element.
   - The function should not print anything with a length less than 1.
   - Use this function to test the arrays generated by your other functions.
2. Design a function called `sum_cubes` that meets the following criteria:
   - The function must accept an array of integers and the number of elements in the array.
   - The function must return the sum of the cubes of all the elements in the array.
   - Example: If an array of length three, containing the values: 2, 4 and 3, the function should return the sum of 8, 64 and 27, which is 99.
3. Design a function called `multiply_all` that meets the following criteria:
   - The function must accept an array of doubles, the number of elements in the array, and a double.
   - The function should update the array by multiplying the additional argument value to every element in the array.
4. Design a function called `sum_below` that meets the following criteria:
   - The function must accept an input array of integers, the number of elements in the input array, an empty output array, the number of elements in the output array, and the threshold value.
   - You may assume that the output array is the same length as the input array.
   - Your function should copy all elements that are below the threshold to the output array. The values should be copied to the output array starting at index 0 in the **reverse order** of how they appear in the input array.
   - Any empty array slots in the output array should be filled with the value -111.
   - The input array should remain unchanged.
   - The function should return the sum of all the numbers copied from the input array to the output array.

Examples:

Function Call	Input Array	Output Array	Returns
`sum_below(input, 5, output, 5, 4)`	[0, 2, 3, 6, 25]	[3, 2, 0, -111, -111]	5
`sum_below(input, 4, output, 4, 1)`	[0, 2, 3, 6]	[0, -111, -111, -111]	0
`sum_below(input, 1, output, 1, 110)`	[5]	[5]	5

5. Design a function called `does_contain_multiples` that meets the following criteria:
   - The function must accept an array of integers, the number of elements in the array, and an integer value as arguments.
   - The function should return 1 if any element in the array are multiples of the final integer argument. Otherwise the function should return 0.
   - Note: It is permissible to copy your is_multiple function from Assignment 3 into this assignment file to use as a helper function for the function `does_contain_multiples`. Alternatively, you must write a new helper function.
6. Design a function called `count_if_contains_multiples` that meets the following criteria:
   - The function must accept an array of integers, the number of elements in the first array, a second array of integers, the number of elements in the second array.
   - The function should return a count of the number of elements in the second array for which the first array contains multiples.
   - You should be making use of at least one of your previous functions as a helper function.
   - Note: The arrays are not guaranteed to be the same length.
   - Example: If the first array contains the values [9, 7, 18, 12, 21] and the second array contains the values [12, 5, 14, 3] the function should return 2. Looking at each element in the second array: a multiple of 12 is in the first array and a multiple of 3 is in the first array and the first array does not contain any multiples of 5 and 14
7. Design a function called `find_min` that meets the following criteria:
   - The function must accept the number of rows, the number of columns in the array, and a 2D array of integers.
   - The function must find and return the minimum value in the 2D array.
8. Design a function called `keep_even` that meets the following criteria:
   - The function must accept the number of rows, the number of columns in the array and an array of integers.
   - The function should update the array so that it is left with only the even values.
   - The function should copy the even values to the start of each row (values should remain in their rows) and they should appear in the order they originated in.
   - Represent deleted values with -111.
   - The function should return the number of the deleted values from the array.