Week 5 Assignment Specification
Module 2 – Pointers and Arrays in C

This document contains the specification for all the assignments of the week.
Each question found here should have the following structure

- Question Preamble or Context (for application questions)
- Your Programming Task
- Input format
- Input constraints (i.e. the range of possible values of inputs to your program)
- Output format
- Starter Files (and how you can find them)
- Test Cases (at least one per question)

Submit to me all the work you have completed and I will help you test and review your code.

There are **8** programming assignments for this week.
3 of these assignments highlighted in yellow, do these first and you can complete the rest of the assignments as you revise in your own time.
1 of these assignments are highlighted in blue, these questions are slightly unique, they are meant to mimic the style of questions that you will find in the Coding Practicals.

Table of Contents

FOREWORD (Contact me if you have any questions!!)
Throughout this module, you will be gradually shifting from the standard compilation process because it may be lengthy to write.

Download the `code_files` folder on the google drive as it is. This will allow you to use the 'make' command to compile your code.

To use the 'make' command, enter your Ubuntu terminal on your computer. In the code_files directory, type in the command:

```
make
```

for instructions on its utility.

1 Left Rotation of Array

Preamble
An array rotation to the left refers to wrapping of the first element of the array to become the last element of the array.

For example, if an array consists of the following elements:

1	2	3	4	5	6	7

1 array rotation would give the following result:

2	3	4	5	6	7	1

3 array rotations **to the initial array** would give this result:

4	5	6	7	1	2	3

Your Task:
Write a program in C that outputs an array A' after n array rotations to the left, given an array A of size s.

Input Format:
The first line should contain the value s.
The second line should contain s integers separated by spaces that make up the array A. The first integer is a_0, the second integer is a_1, and all the way to a_{s-1}, where a_i is an element of the array A $\forall\ 0 \leq i \leq s-1$ where $i\ \in \mathbb{Z}$.
The third line should contain the value n.

Constraints:
$1 \leq s \leq 10^5, -10^6 \leq a_i \leq 10^6\ \forall\ 0 \leq i \leq s-1, 0 \leq n \leq 10^8$.

Output Format:
The array A' with its elements a_i' separated by spaces.

Starter File:
`1_leftrotation.c`
Using this starter code is optional.
This code file implements the input and output formats as specified above, and passes the input to an empty function which you have to implement. The code file would then take the result of your implementation and print it to the console in the specified format.

If you have a different idea in mind, feel free to create a new code file for this question.

[Turn Over for test cases]

TEST CASES FOR Q1

Test Case 1

Input

```
7
1 2 3 4 5 6 7
0
```

Output:

```
1 2 3 4 5 6 7
```

Test Case 2

Input:

```
7
1 2 3 4 5 6 7
7
```

Output:

```
1 2 3 4 5 6 7
```

Test Case 3

Input:

```
7
1 2 3 4 5 6 7
3
```

Output:

```
4 5 6 7 1 2 3
```

END OF Q1

2 Swap Two Doubles using Call By Reference

Preamble
In this question, you will write a C program to swap two doubles using pointers and functions.

Your Task:
Create two variables of type char and name them c1 and c2. Using a function and pointers to these characters, swap the values in c1 and c2 with each other.

Input Format:
The input consists of 1 line, two doubles separated by a space – first the double d_1 followed by the double d_2. Assign the values

Constraints:
For a double d_i, $1.7 \times 10^{-308} \leq d_i \in \mathbb{R} \leq 1.7 \times 10^{308}$ for $i = 1,2$.

Output Format:
On the first line, print the value of d_1. On the second line, print the value of d_2. These two values should have been swapped in the output. Output d_1 and d_2 to **5 decimal places.**

Starter file:
`2_swap.c`
This starter file is as good as empty. Create the solution from scratch. All the best!

TEST CASES FOR Q2
Test Case 1
Input:
```
2.5555 4.00000
```
Output:
```
4.00000
2.55550
```

Test Case 2
Input:
```
-1 -3
```
Output:
```
-3.00000
-1.00000
```

END OF Q2

3 Vending Machine III

Preamble:
Now that you have learnt arrays and pointers, you are tasked to make a new increment your vending machine in new ways.

In the first increment of the vending machine, you created the basic structure of the vending machine and in the second increment, you improved your code's readability, by organizing the code into the following functions.

Your Task:
Make the following changes to your implementation

- Create an array of size 5 to represent the drinks in this vending machine D1 to D5. Store the price of the drinks at the respective indices of this array.

For the following user defined functions,

- Modify User Defined Function #2: `getDrink(…)`
- Modify User Defined Function #4: `getAddMoreMoneyStatus(…)`
- Modify User Defined Function #5: `getValidDrinkNum(…)`

their return type is currently an `int`, but for the sake of our learning about the use of pointers, we will be using pointers to pass information between functions. As such, we will be changing all their respective return types to be `void`.

For the following user defined function,

- Modify User Defined Function #3: `getChange(…)`

you will not change the function's return type. It will still be an integer. However, you will be adding additional functionality to this function. In particular, instead of just returning the smallest number of coins for change, you will also use pointers to return the number of coins of each denomination (i.e. how many $1, how many 50 cents, how many 20 cents, etc.) that constitute this smallest number of coins.

Read the starter file comments for more detailed instructions.

Starter File:
`3_vendingmachine3.c`
You are required to use this starter file. Refer to **Task 1 to Task 6** in the starter file for more information.

TEST CASES FOR Q6
There are no test cases for this question.

END OF Q3

4 Sorting an array of n elements

Preamble:

In Computer Science, selection sort is a simple sorting algorithm. This sorting algorithm divides the array into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire array.

An array of unsorted elements could be:

2	4	5	3	7	1	6

(The unsorted portion of the array would be coloured red.)

The smallest element is then selected from the unsorted array and swapped with the leftmost element, and the first element now becomes part of the sorted array, coloured blue.

For the above array, the element 1 (the minimum in the unsorted portion) is swapped with the leftmost element 2. The resultant array after 1 iteration of the selection sort algorithm would be

1	4	5	3	7	2	6

The selection of the smallest element followed by swapping would be repeated until the entire array is sorted.

Array after the 2nd iteration of selection sort algorithm:

1	2	5	3	7	4	6

Array after the 3rd iteration of selection sort algorithm:

1	2	3	5	7	4	6

Array after the 7th iteration of selection sort algorithm:

1	2	3	4	5	6	7

Your Task

Given an unsorted array A, sort the array A of size n using the selection sort algorithm described above.

Input Format:

The first line is the integer n. The second line consists of n integers separated by spaces, which are the elements of array A, a_0, followed by a_0, then a_1, to a_{n-1}.

Constraints:

$0 \leq n \leq 10^5, -10^8 \leq a_i \leq 10^8 \; \forall \; i \in [\mathbb{Z}: 0, n-1]$.

Output Format:

The sorted array A', with its values separated by spaces.

Starter File:

`4_selectionsort.c`

Using this starter code is optional.

This code file implements the input and output formats as specified above, and passes the input to an empty function which you have to implement. The code file would then take the result of your implementation and print it to the console in the specified format.

If you have a different idea in mind, feel free to create a new code file for this question.

TEST CASES FOR Q4

Test Case 1

Input:

```
5
1 1 1 1 1
```

Output:

```
1 1 1 1 1
```

Test Case 2

Input:

```
5
1 2 3 4 5
```

Output:

```
1 2 3 4 5
```

Test Case 3

Input

```
5
5 4 3 2 1
```

Output:

```
1 2 3 4 5
```

Test Case 4

```
6
0 -10000 -100000 -1000000 -100000000 -10000000
```

Output:

```
-100000000 -10000000 -1000000 -100000 -10000 0
```

END OF Q4

5 Container With Most Water

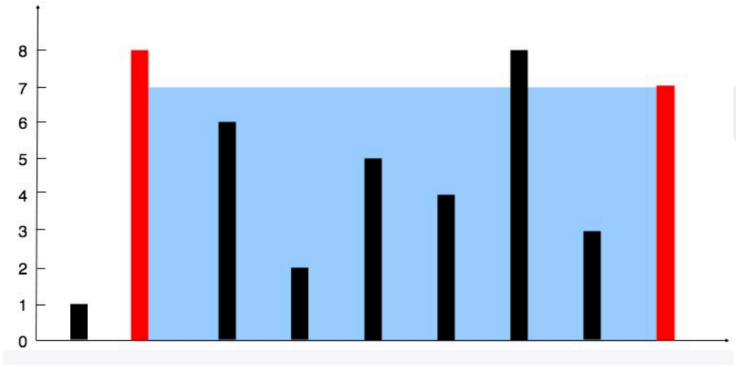
Question adapted from LeetCode.com

Preamble

You are given n non-negative integers $a_1, a_2, \ldots, a_n$, where each represent a point of coordinates (i, a_i). n vertical lines are drawn such that the two endpoints of the line i is (i, a_i) and $(i, 0)$.

Two of these n vertical lines, together with the x-axis, form a two dimensional container – where the x-axis is the base, and the **shorter** of these two vertical lines form the height of the container.

For example, in the diagram below, there are 9 vertical lines, and the container formed with the two lines in red is marked out by the region in blue.



You are **not** allowed to slant the container.

Your Task:

Write a program in C that finds the container with the largest cross-section given the set of n lines.

Input Format:

The first line would contain an integer n. The next line would consists of integers $a_1, a_2, \ldots, a_n$.

Constraints:

$2 \leq n \leq 10, 0 \leq a_i \leq 10^4$ and $n, a_i \in \mathbb{Z} \; \forall \; 1 \leq i \leq n$.

Output Format:

Output only 1 integer – which is the maximum possible volume of water stored in this container.

Starter File:

`5_water.c`

Using this starter code is optional.

This code file implements the input and output formats as specified above, and passes the input to an empty function which you have to implement. The code file would then take the result of your implementation and print it to the console in the specified format.

If you have a different idea in mind, feel free to create a new code file for this question.

[Turn Over for test cases]

TEST CASES FOR Q5

Test Case 1

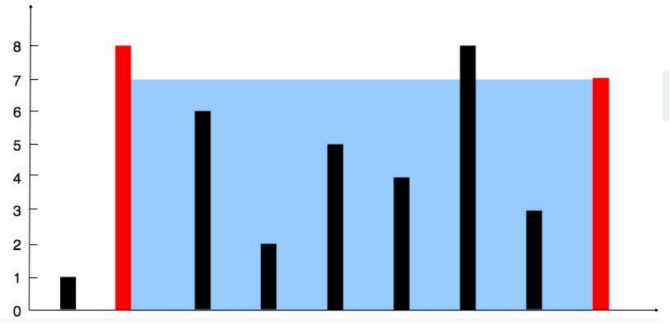
Input:

```
9
1 8 6 2 5 4 8 3 7
```

Output:

```
49
```

Explanation:



The above vertical lines are represented by array [1, 8, 6, 2, 5, 4, 8, 3, 7]. In this case, the max area of water (given by the blue section) the container can contain is 49.

Between the two red lines, the shorter line is 7. So height of cross-section is 7 units. Since the 2 red lines are 7 units (indices) apart, the area is 7 x 7 = 49 units^2.

Test Case 2

Input:

```
2
1 1
```

Output:

```
1
```

Test Case 3

Input:

```
5
4 3 2 1 4
```

Output:

```
16
```

Test Case 4

Input:

```
3
1 2 1
```

Output:

```
2
```

END OF Q5

6 Migratory Birds

Question modified from HackerRank.com using information from perkypet.com

Preamble:
In North America, birds migrate in the late summer through the fall and in the late winter through the spring. Migrations generally occur along a north-south pathway, although a few bird species – namely oceanic birds – may migrate in a circular pattern.

A bird watching organization while gathering data on bird migration, decides to focus on the migration of the following birds (on the left column) during the American fall**,** and they came up with the following table such that each bird species corresponds to a unique species identification number.

Bird Species	Identification Number (ID)
Robin	1
Purple Martin	2
Oriole	3
Bluebird	4
Goldfinch	5
Red-winged Blackbird	6
Mourning Dove	7

It is given that the bird organization is only watching for these specific species of birds.

Your Task:
Given an array of bird sightings where every element represents a bird type ID, determine the most frequently sighted species. If more than one type is spotted at that maximum amount, return the species with the smallest ID.

Input Format:
The first line would be the value n, where n is the size of the array containing bird sightings.
The second line would contain n integers which are the elements of the array $a_1, a_2, \dots, a_n$.

Constraints:
$5 \leq n \leq 2 \times 10^5, 1 \leq a_i \leq 7 \; \forall \; 1 \leq i \leq n$, where a_i is an element of the array of bird sightings.

Output Format:
Output 1 line – which is the **name** of the most frequent species **in all caps**.

Starter File:
`6_birds.c`
Using this starter code is optional.
This code file implements the input format as specified above, and passes the input to an empty function which you have to implement. **The output format is currently incomplete –** it will print the ID number of the most frequent migratory species (instead of the name) if you implement the function correctly. Adapt the code for output to suit the output format specified above.

If you have a different idea in mind, feel free to create a new code file for this question.

[Turn over for test cases]

TEST CASES FOR Q6

Test Case 1

Input:

```
5
1 1 2 2 3
```

Output:

```
ROBIN
```

Explanation:
There are two of bird ID 1 and 2, and one sighting of ID 3. Pick the lower of the two types seen twice: which is bird ID 1. The species of ID 1 is Robin.

Test Case 2

Input:

```
6
1 4 4 4 5 3
```

Output:

```
BLUEBIRD
```

Test Case 3

Input:

```
11
1 2 3 4 5 4 3 2 1 3 4
```

Output:

```
ORIOLE
```

END OF Q6

7 Comparing Two Strings

Attempt this question after watching the additional lecture video on Strings.

Preamble:
As we have formally defined in the additional lecture video on strings, strings are actually an array of type char, with a maximum length determined by the size of the array declared for the string. A string also ends with a sentinel, which is denoted by a `'\0'` character in code.

For instance, a *string* "Hello Xiangle!" can be represented in an array form like this:

'H'	'e'	'l'	'l'	'o'	' '	'X'	'i'	'a'	'n'	'g'	'l'	'e'	'!'	'\0'

An array of 15 characters is required to store the entire string. Also note that the space (at index 6) is also a character with a unique ASCII value.

Lexicographic Equality
Two strings can be said to be **lexicographically equal** if and only if every character in the string is equal in their ASCII value. The order of the ASCII values has to match as well. Refer to the examples below:

The string "HELLO XIANGLE!" and the string "HELLO XIANGLE" is **not lexicographically equal** because the first string has the character '!' at the end and the second does not. The string "Hello Xiangle!" and the string "hello Xiangle!" is also **not lexicographically equal** because the first character of string 1 and string 2 have a different ASCII values ('H' vs 'h').

The string "a" and the second string "a" **are lexicographically equal.** Two empty strings **are lexicographically equal** as well.

Your Task:
Write a C program to check whether two strings, s_1 and $s_{2,}$, are lexicographically equal.

Input Format:
The first line would contain an integer $n,$ a **character** count that s_1 and s_2 will not exceed.
The 2nd line will consist of the integer 1, followed by a space, and followed by the string s_1.
The 3rd line will consist of the integer 2, followed by a space and followed by the string s_2.

Constraints:
You are **not allowed** to use any functions defined in the `string.h` header file.
$2 \leq n \leq 10^8, 0 \leq s_{1_i}, s_{2_j} \leq 255 \; \forall \; i, j \in \mathbb{Z}, i, j \in [0, n-1]$.

Output Format:
If the strings are lexicographically equal, output just 1 line, the digit 1.
If the strings are *not* lexicographically equal, output two lines.
The first line is the digit 0. The second line is the index k where $s_1[0]$ to $s_1[k]$ and $s_2[0]$ to $s_2[k]$ are lexicographically equal. If $s_1[0] \neq s_2[0]$, then the second line should contain the digit -1.

Starter File:

`7_samestring.c`

You are **highly encouraged** to use this code file. Input and output specifics with strings can be tricky. This code file implements the input and output formats as specified above, and passes the input to an empty function which you have to implement. The code file would then take the result of your implementation and print it to the console in the specified format.

TEST CASES FOR Q7

Test Case 1:

Input (Be really precise – copy 1 string at a time)

```
5
1 Hi
2 hi
```

Output:

```
0
-1
```

Explanation:

These two strings are not lexicographically equal. The first character of the sequence isn't equal.

Test Case 2:

Input (Be really precise – copy 1 string at a time)

```
300
1 Lorem ipsum dolor sit amet, his at suas ubique, nonumy labitur vim id,
utinam latine id eam. Mel nihil probatus et, ius sonet euripidis eu. Sea
an utinam consequat, sea laoreet patrioque an. Decore insolens
consectetuer te vis.
2 Lorem ipsum dolor sit amet, his at suas ubique, nonumy labitur vim id,
utinam latine id eam. Mel nihil probatus et, ius sonet euripidis eu. Sea
an utinam consequat, sea laoreet patrioque an. Decore insolens
consectetuer te vis.
```

Output:

```
1
```

Explanation:

These strings are lexicographically equal.

[Turn Over for more test cases]

Test Case 3

Input: (Be really precise, copy 1 string at a time)

```
500
1 Te eam altera recteque, viris blandit te nam. Est id unum nostro
theophrastus, vis accusam persecuti ea. Per fugit nostrud definiebas an.
His at oratio invidunt. At veniam recteque duo, dico hinc inermis et est.
Eos ex regione philosophia, omnium audire usu ea, id pri animal placerat.
Ius ut tale exerci primis, mea no minimum dissentias.
2 Te eam altera recteque, viris blandit te nam. Est id unum nostro
theophrastus, vis accusam persecuti ea. Per fugit nostrud definiebas an.
his at oratio invidunt. At veniam recteque duo, dico hinc inermis et est.
Eos ex regione philosophia, omnium audire usu ea, id pri animal placerat.
Ius ut tale exerci primis, mea no minimum dissentias.
```

Output:

```
0
137
```

Explanation:

At index 138, the first string has a capital letter ‘H’, while the second string has a small letter ‘h’. This makes the second string lexicographically smaller than the first.

END OF Q7

8 Non-Constructible Change

Question adapted from AlgoExpert.io

You need to have done the following before starting:

- Completed Question 4 of this assignment
- Have 1 piece of foolscap paper with you

Time: 30 minutes

Preamble:

You are given a bag of coins of some denominations. A coin i has a denomination d_i.

In this question, you will be writing a function that returns the minimum amount of change (the minimum sum of money) that you **CANNOT** create. You will be given an array of positive integers representing the values of coins in your possession, and the given coins can have any positive integer value d_i.

For instance, you are given coins of 1 cent, 2 cents, and 5 cents, the minimum amount of change that you **cannot create** is 4 cents. If you're given no coins, the minimum amount of change that you can't create is 1.

Below are the implementation steps to find the minimum value of change that you can't return.

1. Create a variable to store the amount of change that you can currently create up to. This variable can be initialised to 0.

2. Sort all your coins using the Selection Sort Algorithm that you implemented in Q2.

3. Loop through your array of coins in ascending order. At every iteration, compare the current coin to the amount of change that you can currently create up to.
   Here are the two scenarios that you'll encounter:
   A) The coin value is **greater than** the amount of change that you can currently create plus 1.
   B) The coin value is **smaller than or equal to** the amount of change that you can currently create plus 1.

Questions

(a) Explain why when case A is encountered, you have found the minimum non-constructible change. [2]

(b) Explain why when case B is encountered for a coin i, the variable you initialised in step 1 increases by the value d_i. [3]

[Turn to the next page for your programming task]

YOUR PROGRAMMING TASK

(c) **Hence,** write a program to find the minimum amount of change that you **cannot create.** [4]

Input format

The first line consists of n, where n is the total number of coins you have.
The second line consists the values $d_0, d_1, d_2, \ldots, d_{n-1}$, where the d_i is the denomination for the i-th coin in the array.

Constraints

$1 \leq n \in \mathbb{Z} \leq 10^3, 1 \leq d_i \in \mathbb{Z} \leq 10^5$.

Output Format

Output the maximum amount of change that you cannot create.

Starter File:

`8_nochange.c`

Using this starter code is optional.
This code file implements the input and output formats as specified above, and passes the input to an empty function which you have to implement. The code file would then take the result of your implementation and print it to the console in the specified format.

TEST CASES FOR Q8

Test Case 1

Input:

```
7
5 7 1 1 2 3 22
```

Output:

```
20
```

Test Case 2

Input:

```
5
1 1 1 1 1
```

Output:

```
6
```

Test Case 3

Input:

```
9
1 5 1 1 1 10 15 20 100
```

Output:

```
55
```

END OF Q8

END OF WEEK 5 ASSIGNMENT