ITCS 201 – Fundamentals of Programming
Lecture 7: Lab Assignments

Q1: Write a program that first receives the size of an array, n (assume that n>0). It then creates an array of size `n`, and asks the user to fill in **`n` decimal numbers**. Next the program uses the array to do the following:

1.1 Compute the average of all elements in the array (with two decimal places).
Sample inputs and outputs:
Case 1:

Input	Output	Expected screen
8 -9 3 5 17 3 0 0 1	2.50	8 -9 3 5 17 3 0 0 1 2.50

Case 2:

Input	Output	Expected screen
1 7.5	7.50	1 7.5 7.50

1.2 Determine the sum of maximum and minimum elements of the array (with two decimal places).

Sample inputs and outputs:

Case 1:

Input	Output	Expected screen
4 1 5 0 -9	-4.00	4 1 5 0 -9 -4.00

Case 2:

Input	Output	Expected screen
3 0 0 0	0.00	3 0 0 0 0.00

Case 3:

Input	Output	Expected screen
7 8 10 0.5 1 6 15 5	15.50	7 8 10 0.5 1 6 15 5 15.50

Q2: Write a program that receives an integer, `n` (assume that `n>0`), and creates two arrays of size `n`. Then it asks the user to fill in **integer numbers** for both arrays. Then, compute and print out the dot-product between the two arrays.

Example: Suppose we have two arrays: `A={1,2,3,4,5}` and `B={8,1,2,3,4}`. The dot-product between these two arrays can be computed as follows:

$A \cdot B = (A_1 \times B_1) + (A_2 \times B_2) + (A_3 \times B_3) + (A_4 \times B_4) + (A_5 \times B_5)$
$= (1 \times 8) + (2 \times 1) + (3 \times 2) + (4 \times 3) + (5 \times 4)$
$= 48$

Sample inputs and outputs:

Case 1:

Input	Output	
5 1 2 3 4 5 8 1 2 3 4	48	5 1 2 3 4 5 8 1 2 3 4 48

Case 2:

Input	Output	
4 -3 2 5 1 3 -7 0 8	-15	4 -3 2 5 1 3 -7 0 8 -15

Q3: Write a program that receives an integer, `n` (assume that `n>0`), and creates an array of size `n`. Then it asks the user to fill in values. Next it asks the user to input two integers, `a` and `b`, **replace all `a` appears in the array to be `b`**, and print the new array. If no `a` appears in the array, print "not found".

Sample inputs and outputs:

Case 1:

Input	Output	Expected screen
5 1 1 **2 2 2** **2 4**	1 1 **4 4 4**	5 1 1 2 2 2 2 4 1 1 4 4 4

Case 2:

Input	Output	Expected screen
7 3 -7 3 1 1 9 -8 0 7	not found	7 3 -7 3 1 1 0 -8 0 7 not found

Case 3:

Input	Output	Expected screen
7 **3** -7 **3** 1 1 0 -8 **3 -7**	**-7** -7 **-7** 1 1 0 -8	7 3 -7 3 1 1 0 -8 3 -7 -7 -7 -7 1 1 0 -8

Q4: Write a program that receives an integer, `n` (assume that `n>0`), and creates an array of size `n`. Ask the user to fill in values **one by one** and allow them to put only zero and one. If they try to add anything other than that, **let them re-input (Hint: data validation)**. Finally, we would have got a binary array. Convert them into a decimal number and print both binary and decimal. Note that we count the first input as the first bit **(Hint: you need to print the binary array in the reverse order).**

Binary to decimal

1010 (binary) = $1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 8 + 0 + 2 + 0 = 10$ (decimal)

10011 (binary) = $1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 8 + 0 + 2 + 0 = 19$ (decimal)

WARNING Data variation might lead the infinite loop situation and that affect PC^2 performance. To avoid that, make sure that you compile and run in your machine before submission. **In this question, only 0 and 1 are allowed.**

Sample inputs and outputs:

Case 1:

Input	Output	Expected screen
5 1 1 0 0 0	0 0 0 1 1 3	5 1 1 0 0 0 0 0 0 1 1 3

Case 2:

Input	Output	Expected screen
5 1 1 0 **2** **5** 1 0	0 1 0 1 1 11	5 1 1 0 2 5 1 0 0 1 0 1 1 11

Note that, for 2^n, you can use a function `pow(2,n)` from the library `math.h`. We have done this before in the previous lab. Do not forget to add `-lm` while compiling your code.