Programming Fundamentals

Programming Day - Week 10

Introduction

Welcome to your favorite day of the week which is programming day🎉. This week, we shall work together to learn and implement new programming concepts.

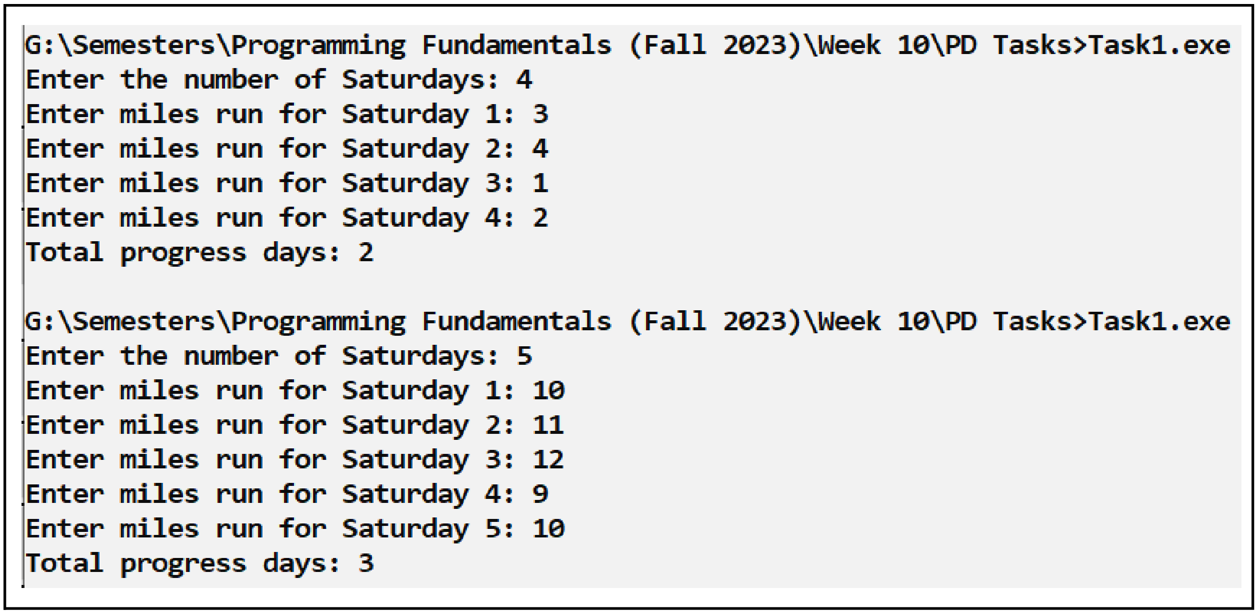
Let's do some coding.

Task 01(CP): To train for an upcoming marathon, Kaka goes on one long-distance run each Saturday. He wants to track how often the number of miles he runs this Saturday exceeds the number of miles run the previous Saturday. This is called a progress day.

Create a program that takes in an array of miles run every Saturday and returns Kaka's total number of progress days.

Test Cases:

Input	Output	Explanation
[3, 4, 1, 2]	2	There are two progress days, (3->4) and (1->2)
[10, 11, 12, 9, 10]	3	
[6, 5, 4, 3, 2, 9]	1	
[9, 9]	0	



```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task1.exe
Enter the number of Saturdays: 4
Enter miles run for Saturday 1: 3
Enter miles run for Saturday 2: 4
Enter miles run for Saturday 3: 1
Enter miles run for Saturday 4: 2
Total progress days: 2

G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task1.exe
Enter the number of Saturdays: 5
Enter miles run for Saturday 1: 10
Enter miles run for Saturday 2: 11
Enter miles run for Saturday 3: 12
Enter miles run for Saturday 4: 9
Enter miles run for Saturday 5: 10
Total progress days: 3
```

Task 02(CP): Declare a String array and take input from the user.

Suppose the user has entered the following data in the array.

```
{"programming", "is", "my", "life", "now"}
```

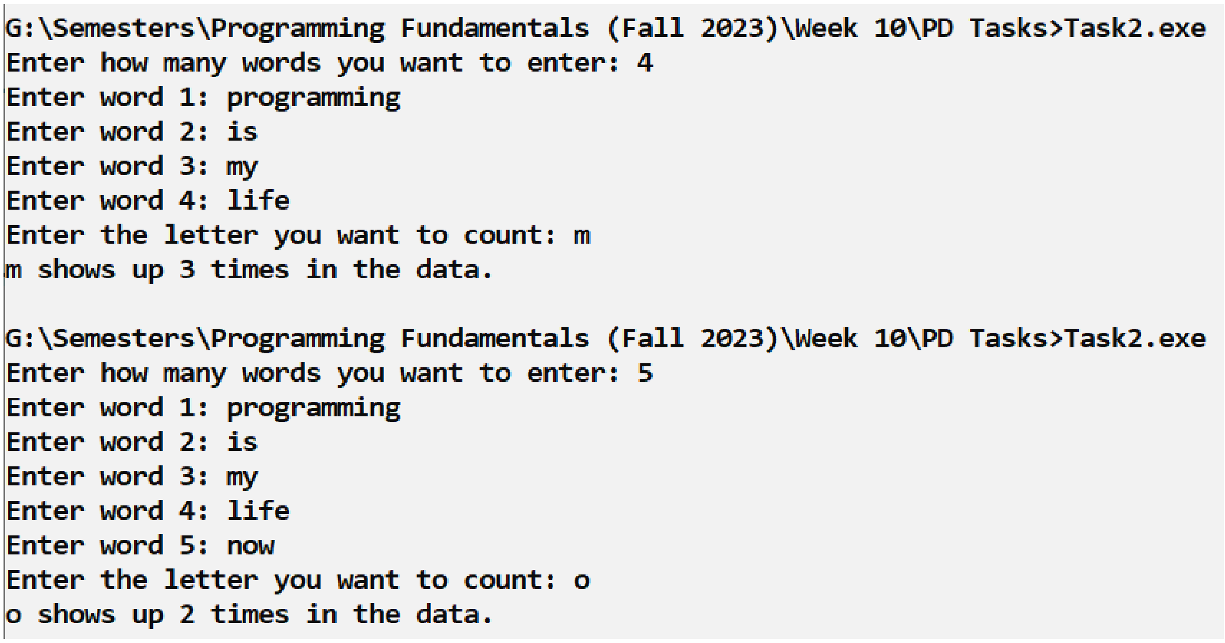
Now, ask the user to Enter a letter.

Suppose the user has entered the letter “o”.

Now your task is to count the number of times a particular letter shows up in the array search.

Test Cases:

Input	**Output**
Enter how many words you want to Enter: 5 Enter word 1: programming Enter word 2: is Enter word 3: my Enter word 4: life Enter word 5: now Enter the letter you want to count: o	o shows up 2 times in the data.
Enter how many words you want to Enter: 5 Enter word 1: programming Enter word 2: is Enter word 3: my Enter word 4: life Enter word 5: now Enter the letter you want to count: m	m shows up 3 times in the data.



```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task2.exe
Enter how many words you want to enter: 4
Enter word 1: programming
Enter word 2: is
Enter word 3: my
Enter word 4: life
Enter the letter you want to count: m
m shows up 3 times in the data.

G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task2.exe
Enter how many words you want to enter: 5
Enter word 1: programming
Enter word 2: is
Enter word 3: my
Enter word 4: life
Enter word 5: now
Enter the letter you want to count: o
o shows up 2 times in the data.
```

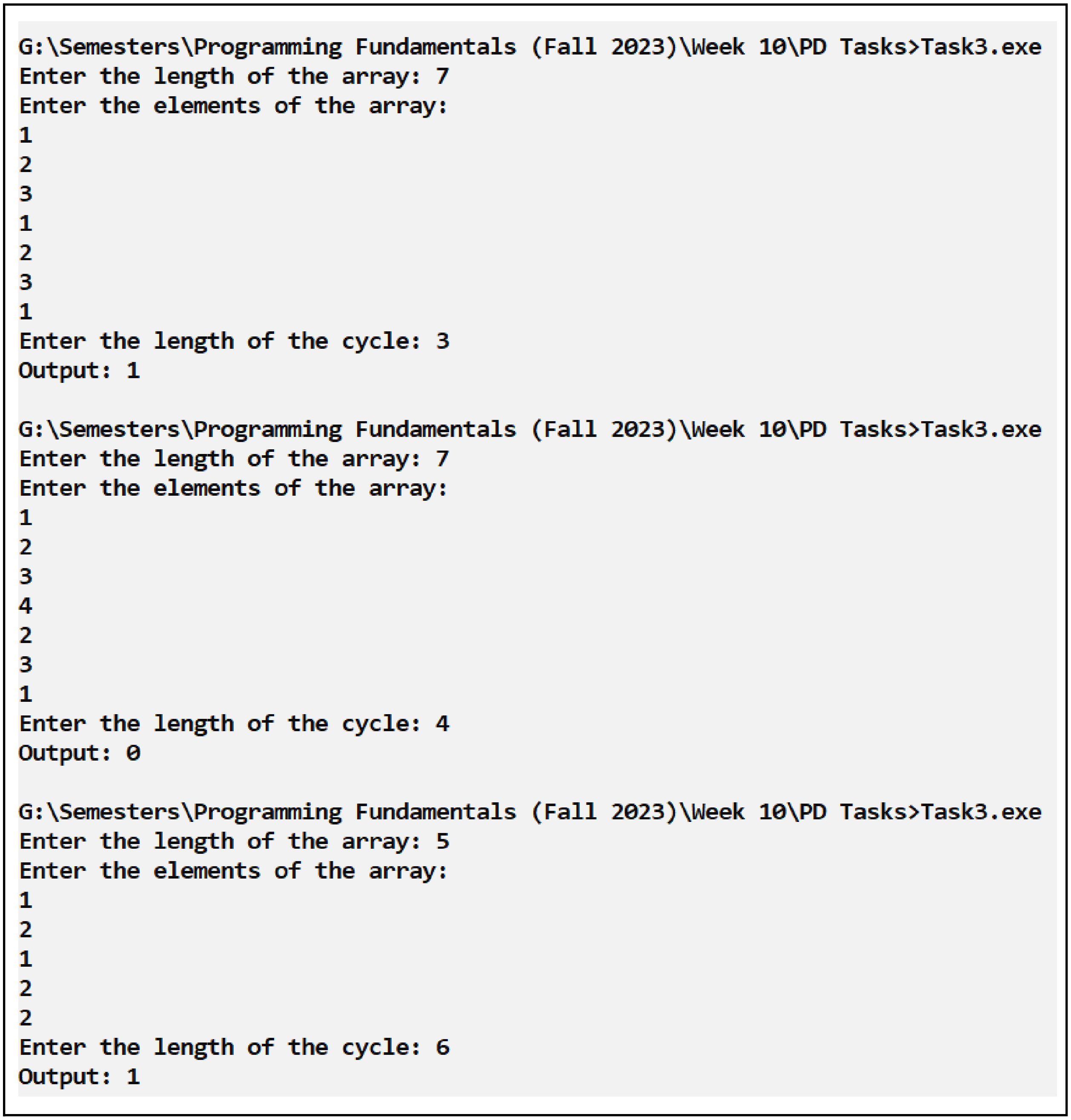
Task 03(CP): Take input from the user in an array and the length of cycle in local variables. You are tasked with writing a function that takes in 1 input as parameter: The length of each cycle.

Return the boolean value true if the array is a repeating cycle, and false if the array is a non-repeating cycle. All cycles begin with the first element of the array. Return true if the cycle length is greater than the array length.

Test Cases:

Input	**Output**	**Explanation**
Array Length: 7 [1, 2, 3, 1, 2, 3, 1] isRepeatingCycle(3)	true	// Since the first two elements of [1, 2, 3] equals [1, 2]
Array Length: 5 [1, 2, 3, 1, 3] isRepeatingCycle(3)	fasle	// Since [1, 2, 3] != [1, 3]
Array Length: 7 [1, 2, 3, 4, 2, 3, 1] isRepeatingCycle(4)	false	
Array Length: 5	true	

[1, 2, 1, 2, 2] isRepeatingCycle(6)		
Array Length: 4 [1, 1, 1, 1] isRepeatingCycle(3)	true	



```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task3.exe
Enter the length of the array: 7
Enter the elements of the array:
1
2
3
1
2
3
1
Enter the length of the cycle: 3
Output: 1

G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task3.exe
Enter the length of the array: 7
Enter the elements of the array:
1
2
3
4
2
3
1
Enter the length of the cycle: 4
Output: 0

G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task3.exe
Enter the length of the array: 5
Enter the elements of the array:
1
2
1
2
2
Enter the length of the cycle: 6
Output: 1
```

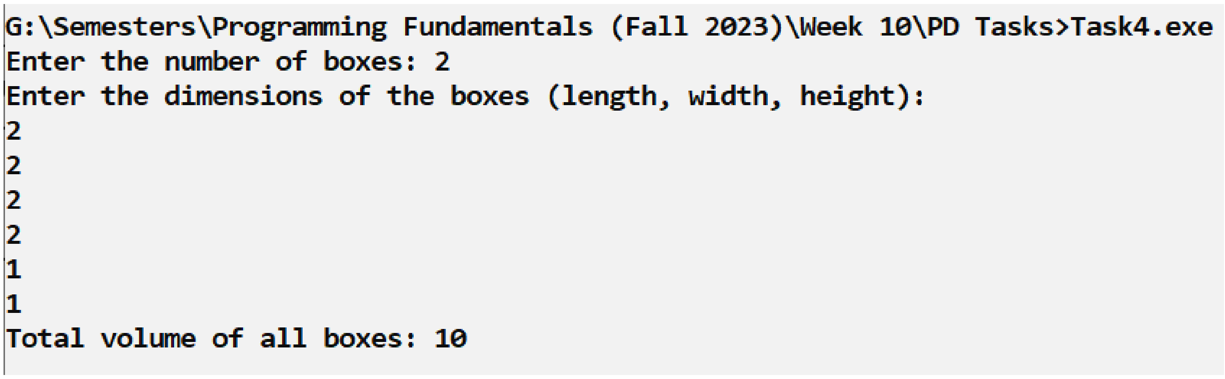
Task 04(CP): Given an array of boxes, create a program that displays the total volume of all those boxes combined together. A box is represented by consecutive 3 elements in the array: length, width and height.
For instance, [2, 3, 2, 6, 6, 7, 1, 2, 1] should display 266 since (2 x 3 x 2) + (6 x 6 x 7) + (1 x 2 x 1) = 12 + 252 + 2 = 266.

Test Cases:

Input	Output
No. of Boxes: 4 [4, 2, 4, 3, 3, 3, 1, 1, 2, 2, 1, 1]	63
No. of Boxes: 2 [2, 2, 2, 2, 1, 1]	10
No. of Boxes: 1 [1, 1, 1]	1

Notes

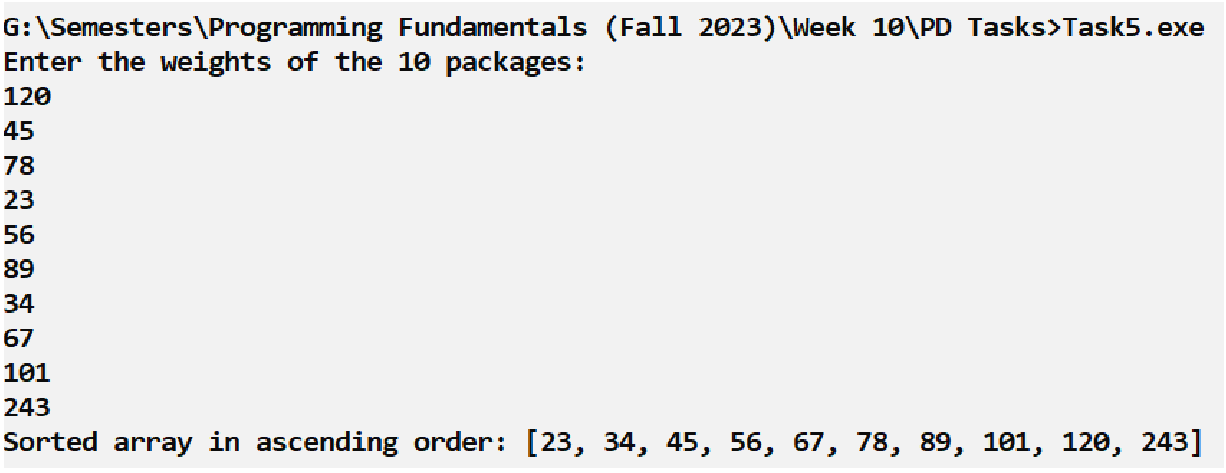
- You will be given at least one box.
- Each box will always have three dimensions included.



```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task4.exe
Enter the number of boxes: 2
Enter the dimensions of the boxes (length, width, height):
2
2
2
2
1
1
Total volume of all boxes: 10
```

Task 05(CP): Imagine you are a warehouse manager and you have a list of 10 packages with their respective weights. Your task is to sort the packages in ascending order of weight so that the lighter packages can be loaded onto delivery trucks first. How would you sort the packages based on their weight in the same array? Write a C++ program to sort the same array in ascending order.

Input	Output
Here is an example of the data in the array of packages: [120, 45, 78, 23, 56, 89, 34, 67, 101, 243]	Sorted array in ascending order: [23, 34, 45, 56, 67, 78, 89, 101, 120, 243]

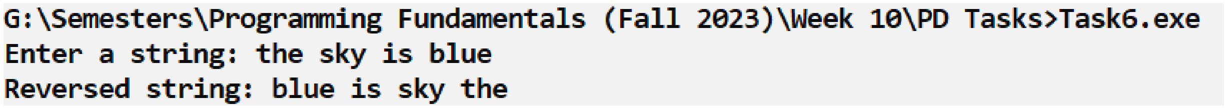


```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task5.exe
Enter the weights of the 10 packages:
120
45
78
23
56
89
34
67
101
243
Sorted array in ascending order: [23, 34, 45, 56, 67, 78, 89, 101, 120, 243]
```

Task 06(CP): Given an input string, reverse the string word by word, the first word will be the last, and so on.

Test Cases:

reverseWords("the sky is blue") ➞ "blue is sky the"
reverseWords("hello world!") ➞ "world! hello"
reverseWords("a good example") ➞ "example good a"



```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task6.exe
Enter a string: the sky is blue
Reversed string: blue is sky the
```

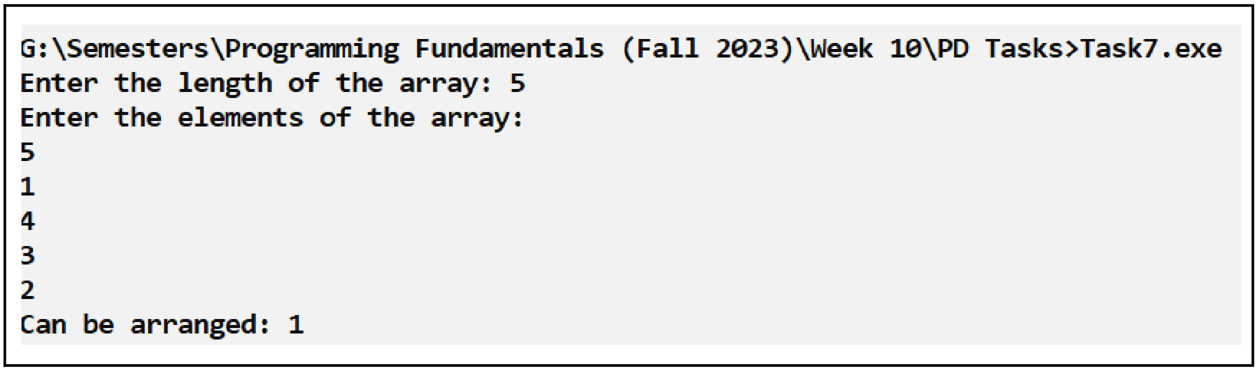
Task 07(CP):

Create a function that determines whether elements in an array can be rearranged to form a consecutive list of numbers where each number appears exactly once.

Test Cases:

Array Length: 5 [5, 1, 4, 3, 2]	**True**	// Can be rearranged to form [1, 2, 3, 4, 5]

Array Length: 6 [5, 1, 4, 3, 2, 8]	**False**	// number 6 and 7 are missing therefore the numbers are not consecutive
Array Length: 6 [5, 6, 7, 8, 9, 9]	**False**	// 9 appears twice



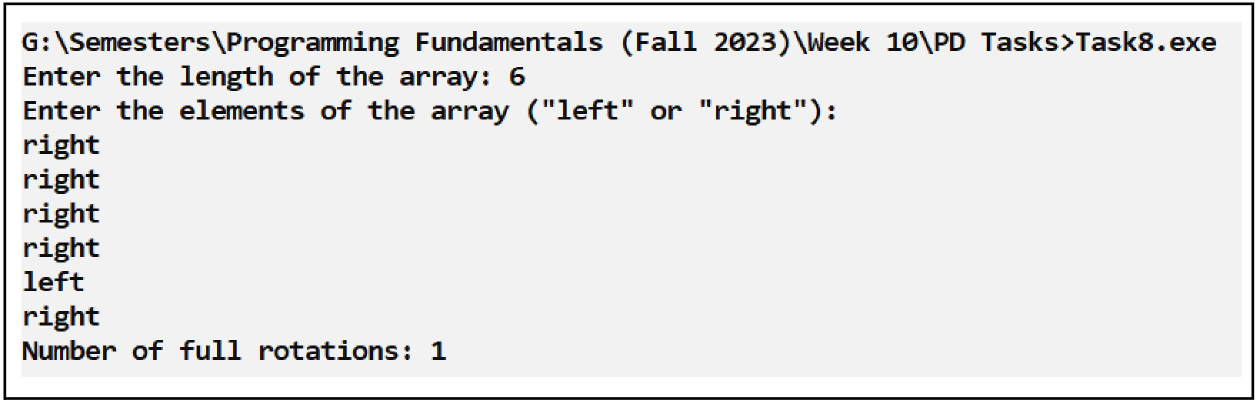
```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task7.exe
Enter the length of the array: 5
Enter the elements of the array:
5
1
4
3
2
Can be arranged: 1
```

Task 08(CP):

Given a list of directions to spin, "left" or "right", return an integer of how many full 360° rotations were made. Note that each word in the array counts as a 90° rotation in that direction.

["right", "right", "right", "right", "left", "right"]	1	// You spun right 4 times (90 * 4 = 360)
		// You spun left once (360 - 90 = 270)
		// But you spun right once more to make a full rotation (270 + 90 = 360)

["left", "right", "left", "right"]	0	
["right", "right", "right", "right", "right", "right", "right", "right"]	2	
["left", "left", "left", "left"]	1	



```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task8.exe
Enter the length of the array: 6
Enter the elements of the array ("left" or "right"):
right
right
right
right
left
right
Number of full rotations: 1
```

Task 09(CP):

You are given an array `nums` of non-negative integers. `nums` is considered **special** if there exists a number `x` such that there are **exactly** `x` numbers in `nums` that are **greater than or equal to** `x`.

Notice that `x` **does not** have to be an element in `nums`.

Return `x` *if the array is* ***special****, otherwise, return* `-1`. It can be proven that if `nums` is special, the value for `x` is **unique**.

Test Cases:

Input: nums = [3,5]

Output: 2

Explanation: There are 2 values (3 and 5) that are greater than or equal to 2.

Input: nums = [0,0]

Output: -1

Explanation: No numbers fit the criteria for x.

If x = 0, there should be 0 numbers >= x, but there are 2.

If x = 1, there should be 1 number >= x, but there are 0.

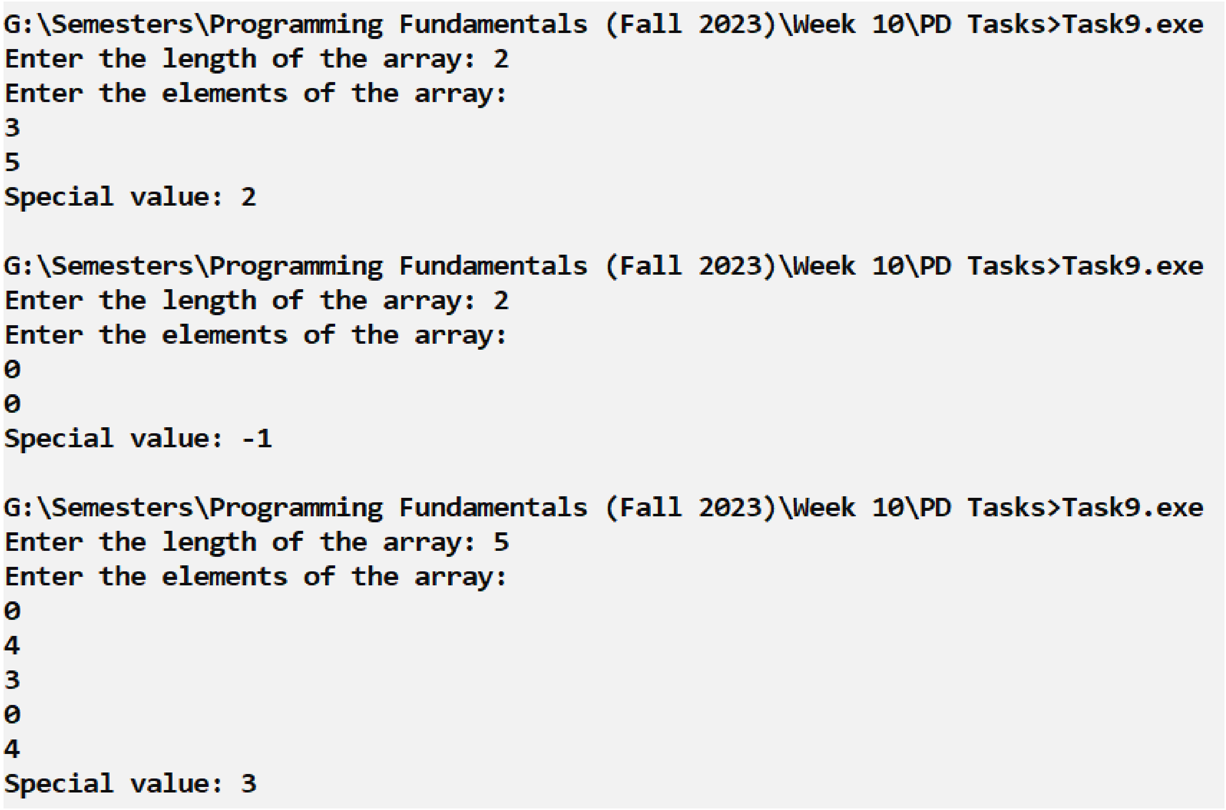
If x = 2, there should be 2 numbers >= x, but there are 0.

x cannot be greater since there are only 2 numbers in nums.

Input: nums = [0,4,3,0,4]

Output: 3

Explanation: There are 3 values that are greater than or equal to 3.



```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task9.exe
Enter the length of the array: 2
Enter the elements of the array:
3
5
Special value: 2

G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task9.exe
Enter the length of the array: 2
Enter the elements of the array:
0
0
Special value: -1

G:\Semesters\Programming Fundamentals (Fall 2023)\Week 10\PD Tasks>Task9.exe
Enter the length of the array: 5
Enter the elements of the array:
0
4
3
0
4
Special value: 3
```