Lab 3 Loop, array

Q1. Loop, array, sorting

You have learnt about the concept of sorting as well as bubble sort during the lecture. Now we consider another approach for sorting. The basic idea is to insert a new element into a sorted subarray during each iteration. If we can put the new element to the suitable position so that the subarray is still sorted, we will finally get the original array in order when we insert all the elements into the subarray.

For example, we have an array as 6, 3, 5, 2, and want to sort it in ascending order using insertion sort. We can go through the following steps to implement the sorting.

1. We start the method by considering the first element as the initial subarray, as we can regard the subarray consist of only one number as sorted. The subarray is now 6;
2. After the initialization, we will insert all the remaining elements to the correct positions in the subarray to keep it sorted. Then we consider the second element 3. It is smaller than 6, so we insert it before 6, and get the updated subarray 3, 6;
3. The next element is 5. It is larger than 3, so we move to the next element in the subarray for comparison. 5 is smaller than 6, which means we find the position for 5 to keep the subarray sorted. And the new subarray is 3, 5, 6;
4. The last remaining element is 2. We find that it is smaller than 3. We insert it before 3, and update the subarray as 2, 3, 5, 6. Now we inserted all the elements into the subarray. The subarray now is in ascending order.

Implement the new sorting method in ascending order, given the size of the array is 6. Output the sorted array.

NOTE: Please try not to create any new array except for the input.

<u>*Expected Outcome:*</u>

Example

```
Enter the element in the array:
6
4
7
2
10
5
The sorted array is:
2, 4, 5, 6, 7, 10
```

Q2. Loop and array

Write a program to find the maximum number among the right-hand side of an element that is smaller than the current element for each element in an integer array. The output of the program is another integer array created by yourself. Assign the value of the maximum value or -1, if the next smaller elements don't exist, at the same position in the output array.

For example: the input array is [5, 6, 3, 4, 1, 2].
We consider the second element 6. The elements that are at the right-hand side of 6 and smaller than it are [3, 4, 1, 2]. Then the maximum value among them is 4. Therefore, at the position of 2 in the output array, we should assign the value of 4.
With the rule, we can get that the output array of [5, 6, 3, 4, 1, 2] should be [4, 4, 2, 2, -1, -1].

The program asks the user to input the size of the array first, followed by every element in the array. Every element of the array is an integer.

Expected Outcomes:

Example 1

```
Enter the size of the array:
6
Enter the elements of the array:
5
6
3
4
1
2
The output is:
4, 4, 2, 2, -1, -1
```

Example 2

```
Enter the size of the array:
6
Enter the elements of the array:
-3
-1
-2
0
-1
-5
The output is:
-5, -2, -5, -1, -5, -1
```

Hint: The size of the input array is not larger than 20. You may create an array first and store the input in it, then use the size to design the loop.

Q3. Loop

Write a program to produce a square matrix with 0's down the main diagonal, 1's in the entries just above and below the main diagonal, 2's above and below that, etc.

```
0 1 2 3 4
1 0 1 2 3
2 1 0 1 2
3 2 1 0 1
4 3 2 1 0
```

Expected Outcomes:

Example 1	Example 2

`Enter the number of rows:`
`5`
`0 1 2 3 4`
`1 0 1 2 3`
`2 1 0 1 2`
`3 2 1 0 1`
`4 3 2 1 0`
	`Enter the number of rows:`
	`8`
	`0 1 2 3 4 5 6 7`
	`1 0 1 2 3 4 5 6`
	`2 1 0 1 2 3 4 5`
	`3 2 1 0 1 2 3 4`
	`4 3 2 1 0 1 2 3`
	`5 4 3 2 1 0 1 2`
	`6 5 4 3 2 1 0 1`
	`7 6 5 4 3 2 1 0`

Example 3	Example 4

`Enter the number of rows:`
`0`
`Please enter positive integer.`
	`Enter the number of rows:`
	`3`
	`0 1 2`
	`1 0 1`
	`2 1 0`

Hint: Consider using nested **for-loop**, with the outer **for-loop** responsible for each row and the inner **for-loop** responsible for each column.