Lab Exam - 1

Instructions

- There are 4 questions, out of which you have to attempt **only 3**.
- For q1, q2 and q3, you will be given two versions of varying difficulty.
- The different versions will carry different marks according to the difficulty.
- You need to solve **exactly one** version for each of the problems.

Question 1

You are given an integer array `nums` and its length `n`.

Constraints:

```
1 <= nums.length <= 100
1 <= nums[i] <= 100
```

Version 1.1 (20 marks)

Create a function `sum` that takes `nums` and `n` as input and returns the sum of all the elements in `nums`.

Declaration of the function `sum`:

```
int sum(int nums[], int n);
```

Example:

```
Input: nums = [1, 2, 3, 4, 5], n = 5
Output: 15
```

Version 1.2 (30 marks)

Create a function `uniqueSum` that takes `nums` and `n` as input and returns the sum of all the unique elements in `nums`. The unique elements of an array are the elements that appear **exactly once** in the array.

Declaration of the function `uniqueSum`:

```
int uniqueSum(int nums[], int n);
```

Example 1:

```
Input: nums = [1, 2, 3, 4, 2], n = 10
Output: 8
Explanation: The unique elements are [1, 3, 4], and their sum is 8.
```

Example 2:

```
Input: nums = [5, 5, 5, 5, 5], n = 5
Output: 0
Explanation: There are no unique elements, so the sum is 0.
```

Question 2

You are given an integer array `nums` and its length `n`.

Constraints:

```
1 <= nums.length <= 1000
-1000 <= nums[i] <= 1000
```

Version 2.1 (20 marks)

Create a function `isNonDecreasing` that takes `nums` and `n` as input and returns `1` if the array is non-decreasing, and `0` otherwise.

An array `nums` is non-decreasing if for all `i <= j`, `nums[i] <= nums[j]`.

Declaration of the function `isNonDecreasing`:

```
int isNonDecreasing(int nums[], int n);
```

Example 1:

```
Input: nums = [5], n = 1
Output: 1
Explanation: The array is non-decreasing.
```

Example 2:

```
Input: nums = [1, 2, 3, 4, 0], n = 5
Output: 0
Explanation: The array is not non-decreasing because nums[3] > nums[4] (4 > 0).
```

Version 2.2 (30 marks)

Create a function `longestNonDecreasingSubarray` that takes `nums` and `n` as input and returns the length of the longest non-decreasing subarray in `nums`.

A subarray is a contiguous part of an array. For example, `[5, 1, 5]` is a subarray of `[2, 5, 1, 5, 7]`.

Declaration of the function `longestNonDecreasingSubarray`:

```
int longestNonDecreasingSubarray(int nums[], int n);
```

Example 1:

```
Input: nums = [5], n = 1
Output: 1
Explanation: The longest non-decreasing subarray is [5].
```

Example 2:

```
Input: nums = [1, 2, 2, 0, -1, 7, 8, 8, 10], n = 9
Output: 5
Explanation: The longest non-decreasing subarray is [-1, 7, 8, 9, 10].
```

Question 3

You are given an integer array `nums` containing `n` distinct integers in the range [0, n]. Print the only number in the range that is missing from the array.

Constraints:

- n == nums.length
- 1 <= n <= 10^4
- 0 <= nums[i] <= n
- All the numbers of nums are unique.

Example 1:

```
Input: nums = [5, 0, 1, 3, 2], n = 5
Output: 4
Explanation: n = 5, so in the range [0, 5], 4 is the missing number since it does not appear in nums.
```

Version 3.1 (20 marks)

The solution should be of time complexity `O(n²)` and space complexity `O(1)`.

Version 3.2 (30 marks)

The solution should be of time complexity `O(n)` and space complexity `O(1)`.

Question 4

(30 marks)

You are given an integer array `nums` and its length `n`.

Create a function `bubbleSort` that takes `nums` and `n` as input and sorts the array in non-decreasing order using the bubble sort algorithm.

After calling the `bubbleSort` function, print the sorted array.

Declaration of the function `bubbleSort`:

```
void bubbleSort(int nums[], int n);
```

Example 1:

```
Input: nums = [5, 1, 4, 2, 8], n = 5
Output: 1 2 4 5 8
Explanation: The array is sorted in non-decreasing order.
```

Constraints:

```
1 <= nums.length <= 1000
-1000 <= nums[i] <= 1000
```