Assignment 06

Instructions

1. Write your functions according to the signature provided below in a python(.py) file and submit them to the autograder for evaluation.
2. The autograder has a limited number of attempts. Hence, test it thoroughly before submitting it for evaluation.
3. Save your python file as text(.txt) file and submit it to the canvas.
4. If the assignment has theoretical questions, upload those answers as a pdf file on the canvas.
5. Submission on the canvas is mandatory and the canvas submitted text file should be the same as that of the final submission of the autograder. If not, marks will be deducted.
6. No submission on canvas or no submission on autograder then marks will be deducted.
7. Access the auto grader at https://c200.luddy.indiana.edu.

Question 1

Given two strings s and t of lengths m and n respectively, return the minimum window substring of s such that every character in t (including duplicates) is included in the window. If there is no such substring, return the empty string "".

Examples

Example 1:

```
Input: s = "ADOBECODEBANC", t = "ABC"
Output: "BANC"
```

Explanation: The minimum window substring "BANC" includes 'A', 'B', and 'C' from string t.

Example 2:

```
Input: s = "a", t = "a"
Output: "a"
```

Explanation: The entire string s is the minimum window.

Example 3:

```
Input: s = "a", t = "aa"
Output: ""
```

Explanation: Both 'a's from t must be included in the window. Since the largest window of s only has one 'a', return empty string.

Constraints

- s and t consist of uppercase and lowercase English letters.
- Time complexity should be atmost $O(M + N)$ where $m = len(s)$ and $n = len(t)$

Function

```
def minWindow(s, t):
  # Your Implementation here
  return 0
```

Question 2

Given an unsorted array of integers nums, return the length of the longest consecutive elements sequence.

You must write an algorithm that runs in O(n) time.

Examples

Example 1:

```
Input: nums = [100,4,200,1,3,2]
Output: 4
```

Explanation: The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.

Example 2:

```
Input: nums = [0,3,7,2,5,8,4,6,0,1]
Output: 9
```

Constraints

- Time complexity should be atmost $O(N)$ where $N = len(nums)$

Function

```
def longestConsecutive(nums)::
  # Your Implementation here
  return 0
```

Question 3

Implement the Instagram class as per the given structure. In Instagram, people can share photos, follow or unfollow others, and see the ten latest posts from the people they follow.

Examples

```
insta = Instagram(3)
insta.sharePhoto(1, 6)
insta.sharePhoto(1, 4)
insta.getFeed(1) # [4, 6]
insta.follow(1, 2)
insta.sharePhoto(2, 3)
insta.sharePhoto(1, 7)
insta.sharePhoto(2, 5)
```

```
#[5, 7, 3, 4, 6] (Photo IDs must be ordered from most recent to least recent).
insta.getFeed(1)
insta.unfollow(1, 2)
insta.getFeed(1) # [7, 4, 6]
```

Constraints

- All the photos have unique IDs.
- Use hashing (Python Dict, Set, etc.) to solve the question.

Function

```
class Instagram:

    # Initialize your Instagram object.
    def __init__(self):
      pass

    # Shares a new photo with the ID photoId of the user userId.
    def sharePhoto(self, userId: int, photoId: int) -> None:
      pass

    # Retrieves the 10 most recent photo IDs in the user's feed.
    # Each photo in the feed must be shared by users who the user
    # followed or by the user themself.
    # Photo IDs must be ordered from most recent to least recent.
    def getFeed(self, userId: int) -> list[int]:
      pass

    # The user with ID followerId started following the user with ID followeeId.
    def follow(self, followerId: int, followeeId: int) -> None:
      pass

    # The user with ID followerId started unfollowing the user with ID followeeId.
    def unfollow(self, followerId: int, followeeId: int) -> None:
      pass
```

Question 4

Welcome to the realm of subarray sorcery! You've been bestowed with an array of integers nums and a mystical integer k. Your quest? To unveil the total number of subarrays whose sum matches the magical number k.

Examples

Example 1:

```
Input: nums = [1, 2, 3, 2, 1], k = 3
Output: 3
```

```
Explanation: Subarrays with sum 3: [1, 2], [3], [2, 1]
```

Constraints

- Solve the question in O(n) time complexity.

Function

```python
def countSum(nums: list[int], k: int) -> int:
    pass
```

Question 5

Given a string s, divide it into as many contiguous substrings as possible, such that no two distinct letters within the substring overlap across different substrings. Return the lengths of these substrings in a list.

Examples

Example 1:

```
Input: ipStr = "abcabcakghh"
Output: [7, 1, 1, 2]
```

Explanation: [“abcabca”,“k”,“g”,“hh”] As we can see that no character repeats in the substring this is the optimal answer.

Constraints

- Solve the question using hashmaps.

Function

```python
def uniqSubstr(ipstr:str) -> list(int):
    pass
```

Question 6

Now it is time for simulations as we end the course. Write the Python function which takes `n` as input and generates a point whose coordinates lie between `-1` and `1` for `n` number of times. You can use the random library of Python to generate the values. Out of these `n` points find the count of points whose value of $x^2 + y^2$ is less than 1 and assign it to the *count* variable.

1. Report the value of $4count/n$ for different values of n.
2. Using libraries of Python plot the n on the x-axis and $4 * count/n$ on the y-axis for different values of n
3. Write your observations for the above graph.

Note: Please submit the answer to this question in a PDF file.

The answer in the PDF file should contain the following,

1. Python code that is used to generate the table and graph.
2. A table containing n and $4count/n$ values
3. Graph for the above table.
4. Observations regarding the same.