Assignment 1

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

In the enchanting world of Bollywood, you're handed a sorted script of romantic tales, where the characters are represented by positive integers portraying their charm. Your mission is to discover two characters whose combined allure matches a specific target. Also, there is a special condition to this allure that one of the charm should be as small as possible and one should be as big as possible. These characters are identified by their unique script positions: character1 at position index1 and character2 at position index2 (where 0 <= index1 < index2 <= total characters - 1). Your task is to help the producer by completing the bollywoodCharm(scripts,expectedCharm) function and select the characters by returning the indexes of the characters as [index1,index2].

Examples

Example 1:

```
Input: scripts = [1, 2, 3, 4, 5], expectedCharm = 9
Output: [3,4]
```

Explanation: The expected charm can only be achieved by using integers at index 3 and 4.

Example 2:

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

Explanation: The expected charm can be achieved by indices [0,5], [1,4] and [2,3] but the once that are farthest is the answer.

Constraints

- The problem must be solved in O(n). where n is the number of tales.
- It is guaranteed that the combined charm would fit in an integer.
- It is also guaranteed that there is atleast one solution for the given scripts.

Function

```
def bollywoodCharm(scripts: List[int],expectedCharm: int)->List[int]:
  # Expects index1 and index2 as a list [idx1,idx2]
  return [idx1,idx2]
```

Question 2

In the enigmatic landscape of genetic evolution, researchers grapple with the intricate task of determining if Gene G1 has emerged as a descendant of Gene G2. For this the scientists check the Gene's proteins and if a sequence of Gene G1 is in Gene G2 (but the order may vary) then they can conclude that the Gene G2 emerged from Gene G1. Your task is to help genealogist by writing a function checkIsGeneDerived(G1,G2) and return a Boolean to indicate that whether Gene G2 is derived from G1.

Examples

Example 1:

```
Input: G1 = "abc", G2 = "popllscabsijjasoidj"
Output: True
```

Explanation: The variation of the gene G1 exists from indexes 6 to 8.

Example 2:

```
Input: G1 = "popllscabsijjasoidj", G2 = "abc"
Output: False
```

Explanation: No variation of G1 exists in G2.

Example 3:

```
Input: G1 = "abc", G2 = "abc"
Output: True
```

Explanation: The variation of Gene G1 exists in gene G2 from index [0,2].

Constraints

- The problem must be solved in O(n).
- 1<=G1.length<=G2.length

Function

```
def checkIsGeneDerived(G1: str,G2: str)->bool:
  # Expects a boolean indicating if G1's variation present in G2
  return isVariant
```

Question 3

In the arsenal of market dynamics, behold an array of missile valuations (valuations) and each index representing the warhead's cost on a designated day and the destruction it can cause is same

as the valuation on that day. Your mission: to optimize your gains by pinpointing a solitary day for missile procurement and selecting another future day for its strategic launch. Your ultimate goal is to complete the function cheapestWarhead(valuations) and help the general to buy a warhead on a day such that the cost of acquisition is the cheapest and fire it on a day so that it causes the maximum damage that is the difference between the damage done and the cost of acquiring is maximum.

Examples

Example 1:

```
Input: valuations = [7,1,5,3,6,4,1,2,3]
Output: 5
```

Explanation: You can buy the warhead on day 2 and fire it on day 5. because the damage on day 5 is 6 and the price on day 2 is 1 making the difference as maximum (5).

Example 2:

```
Input: valuations = [10,9,4,2,1,1]
Output: 0
```

Explanation: Here there is no day you can buy the warhead on a previous day and fire it on a later day at a greater damage hence the difference at any index would be negative.

Constraints

- The problem must be solved in O(n).

Function

```
def cheapestWarhead(valuations: List[int])->int:
  # Expects integer depicting the maximum difference between aquisition and damage.
  # If not possible 0.
  return maxdamage
```

Question 4

Prove or disprove the following questions on asymptotic notation. Provide detailed explanations to convey your ideas clearly.

You can either solve them over paper, scan and upload the pdf file over canvas.
Or
Type the solutions using latex and upload the compiled pdf file over canvas.

a. Prove or disprove that $f(n) \in \Theta(g(n))$ where $f(n) = nlog(n^3)$ and $g(n) = nlogn$

b. Prove or disprove that $f(n) \in \Omega(g(n))$ where $f(n) = (1/3)n^2 + 10n - 2$ and $g(n) = n^2$

c. Prove or disprove that $f(n) \in O(g(n))$ where $f(n) = n^3$ and $g(n) = 100n^2 + 10^6n$

d. Prove or disprove that $f(n) \in o(g(n))$ where $f(n) = \sqrt{n^2 + 1}$ and $g(n) = n^2 + 3$

e. Prove or disprove that $f(n) \in \omega(g(n))$ where $f(n) = n^2 \sin(1/n) + 2n$ and $g(n) = n^2$

Question 5

Find the time complexities of the following functions. Provide detailed solutions.

a.

```
def function1(n):
  for i in range(0, n):
    j = 1
    while j <= i:
      j = i + j
  return
```

b.

```
def function2(n):
  while n > 1:
    n = n + 20
    n = n - 10
    n = n - 15
  return
```

c.

```
def function3(n):
    i = 1
    while i < n:
      i = i * 2
    return
```

d.

```
def function4(n):
  for i in range(0, n ** 2):
    j = n
    while j != 0:
      j = j // 2
  return
```

e.

```
def function5(n):
  i = 2
  while i < n:
    i = i * i
  return
```

Question 6

In the land of Codeville, there lived a brilliant programmer named Nobita. One day, as he was navigating through the labyrinth of data structures, he stumbled upon a Linked List cursed by a mysterious cycle. Filled with despair, Nobita called upon his digital companion, Doraemon, the futuristic cat with a pocket full of magical gadgets.

"Nobita, my friend, fear not! I have just the gadget to help you unravel this tangled web," exclaimed Doraemon as he pulled out the "Cycle-Buster 3000" from his magical pocket.

Your task is to complete the working code of "Cycle-Buster 3000" with given constraints.

Examples

Example 1:

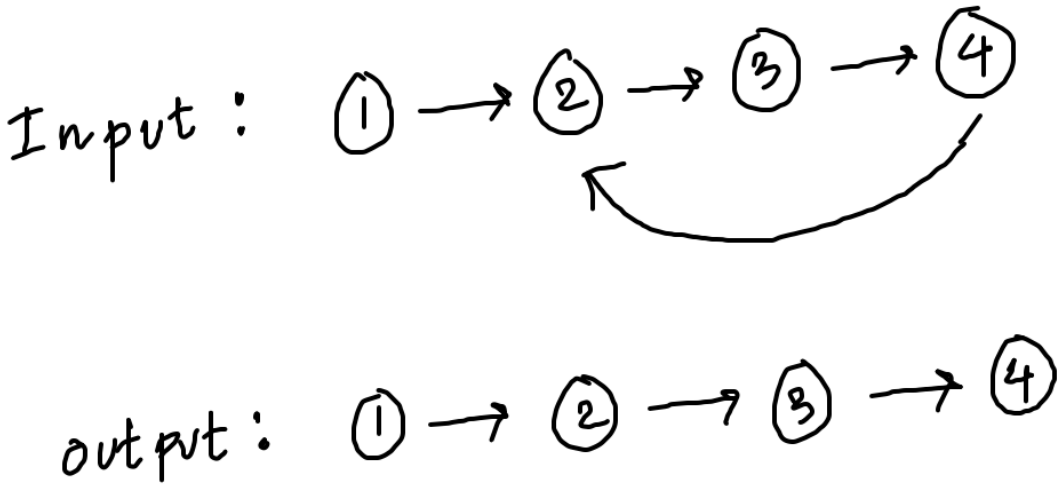

Figure 1: Linked list

Explanation: Head of the linked list is given, There is a cycle in the linked list where tail connects to the second node. Remove this connection so that there will not be any cycle. And return the head of changed linked list.

Constraints

- The problem must be solved in $O(n)$ Time and $O(1)$ Space.

Function

```
class Node:
  def __init__(self, val):
    self.next = None
    self.data = val

def cycle_buster(head):
  return None
```

Question 7

In a charming town, n vertical pillars stood tall, each represented by an integer in the array 'pillarsHeight'. The townsfolk were eager to find two pillars that, when combined with the x-axis, formed a container capable of storing the most water. The task was to identify the perfect pair of

pillars that maximized the container's water-holding capacity. Now, dear reader, can you discover the optimal pair of pillars to maximize water storage?

Examples

Example 1:

```
Input: pillarsHeight = [1, 5, 9, 6, 10, 2, 3]
Output: 18
```

Explanation: Choose pillars with heights of 9 and 10 to store the maximum amount of water.

Example 2:

```
Input: pillarsHeight = [1, 5, 6, 3, 4]
Output: 12
```

Explanation: Choose pillars with heights of 5 and 4 to store the maximum amount of water.

Constraints

- n == pillarsHeight.length
- pillarsHeight[i] >= 0
- Solve the problem in O(n) time complexity.

Function

```python
def maxWaterStorage(pillarsHeight: list[int]) -> int:
    pass
```

Question 8

In a magical land, there was a special string called 's' made up of only '1' and '2.' The magic happened when you counted how many times '1' and '2' appeared consecutively, and this pattern created the string itself.
The string started with "1221121221221121122…" and formed groups like "1 22 11 2 1 22 1 22 11 2 11 22 …", with the number of '1's or '2's in each group following the magical sequence "1 2 2 1 1 2 1 2 2 1 2 2 …".
In this magical land, the people were curious about a simple question: How many times does '1' appear in a special string of length 'n'? Can you solve this enchanting riddle and find the answer?

Examples

Example 1:

```
Input: n = 5
Output: 3
```

Explanation: Special string of length 5 is '12211', where occurance of '1' is 3.

Example 2:

```
Input: n = 8
Output: 4
```

Explanation: Special string of length 8 is '12211212', where occurance of '1' is 4.

Constraints

- n >= 1

Function

```
def specialString(n: int) -> int:
    pass
```

Question 9

In a quaint village, there lived a group of friends who loved playing with letters. They had a magical string called 'inputStr,' consisting of lowercase English letters, and an array 'moves' that held secret instructions. The magic began when they performed a move() on a letter, transforming it into the next letter in the alphabet, with 'z' wrapping around to become 'a.' For instance, move('a') became 'b', move('t') turned into 'u', and move('z') playfully transformed back into 'a.'

Curiosity led the friends to a set of mysterious instructions. For each moves[i] = x, they were to shift the first i + 1 letters of 'inputStr' x times. The village was abuzz with excitement as the friends embarked on this enchanting journey of shifting letters. As the moves were executed one by one, the string 'inputStr' transformed, creating a final string after all the shifts were applied. The villagers were intrigued, and the question that lingered in the air was: Can you, too, unveil the final string after these mystical shifts?

Examples

Example 1:

```
Input: inputStr = "abc", moves = [3,4,7]
Output: "omj"
```

Explanation: We start with "abc". After shifting the first 1 letters of inputStr by 3, we have "dbc". After shifting the first 2 letters of inputStr by 4, we have "hfc". After shifting the first 3 letters of inputStr by 7, we have "omj", the answer.

Constraints

- moves.length == inputStr.length
- moves[i] >= 0
- Solve the problem in O(n) time complexity.

Function

```
def shiftingCharacters(inputStr: str, moves: list[int]) -> str:
    pass
```