**Programming and Problem Solving**

**Assignment # 1**

**PART 1**

**SUBMITTED BY**

**Name:** Manan Dineshbhai Paruthi

**Student ID:** 40192620

**Pseudo Code - Question 1**

**INPUT** arr

**INITIALIZE** middleIndex = sizeOfArr/2

**IF** arr.length%2 != 0 **THEN**

**IF** middleIndex%2 != 0 **THEN**

leftEndIndex = middleIndex

rightStartIndex = middleIndex+2

**ELSE**

leftEndIndex = middleIndex

rightStartIndex = middleIndex+1

**ENDIF**

**ELSE**

**IF** middleIndex%2 != 0 **THEN**

leftEndIndex = middleIndex

rightStartIndex = middleIndex+1

**ELSE**

leftEndIndex = middleIndex

rightStartIndex = middleIndex

**ENDIF**

**ENDIF**

**FOR** i = 0 to leftEndIndex ; **INCREMENT** by 2

temp = arr[i]

arr[i] = arr[i+1]

arr[i+1] = temp

**END FOR**

**FOR** i = rightStartIndex to arr.length ; **INCREMENT** by 2

temp = arr[i] + arr[i+1]

arr[i+1] = temp

**END FOR**

**OUTPUT** arr

1. **Time Complexity:** O(n)
2. **Space Complexity:** O(1)

**Pseudo Code - Question 2**

**INPUT** str

**INITIALIZE**

highestOccurringChar = str.charAt(0)

highestOccurringCharCount = 0

currentChar = str.charAt(0)

currentCharCount = 1

tempStr = “”

**FOR** i = 1 to str.length() ; **INCREMENT** by 1

**IF** str.charAt(i) == currentChar **THEN**

currentCharCount++

**ELSE**

//adding current char to the result

tempStr += Character.toString(currentChar)

**IF** currentCharCount > 1 **THEN**

tempStr += Integer.toString(currentCharCount)

**END IF**

//finding highest occurring character count and the character

**IF** currentCharCount > highestOccurringCharCount **THEN**

highestOccurringCharCount = currentCharCount

highestOccurringChar = currentChar

**END IF**

//reset count for new char

currentChar = str.charAt(i)

currentCharCount = 1

**END IF**

//if last element then add to result

**IF** i == str.length()-1 **THEN**

tempStr += Character.toString(currentChar)

**IF** currentCharCount > 1 **THEN**

tempStr += Integer.toString(currentCharCount)

**END IF**

**END IF**

**END FOR**

**OUTPUT** tempStr

**OUTPUT** highestOccurringChar , highestOccurringCharCount

1. **Time Complexity:** O(n)
2. **Space Complexity:** O(1)

**Pseudo Code - Question 3**

**INPUT** arr

**INITIALIZE**

smallestDiff = Integer.MAX\_VALUE

firstIndexOfSmallestDiff = 0

secondIndexOfSmallestDiff = 0

largestDiff = Integer.MIN\_VALUE

firstIndexOfLargestDiff = 0

secondIndexOfLargestDiff = 0

**FOR** i = 0 to arr.length-1 ; **INCREMENT** by 1

diffBtwTwoConsEle = Math.abs(arr[i] - arr[i+1])

**IF** diffBtwTwoConsEle <= smallestDiff **THEN**

smallestDiff = diffBtwTwoConsEle

firstIndexOfSmallestDiff = i

secondIndexOfSmallestDiff = i+1

**END IF**

**IF** diffBtwTwoConsEle >= largestDiff **THEN**

largestDiff = diffBtwTwoConsEle

firstIndexOfLargestDiff = i

secondIndexOfLargestDiff = i+1

**END IF**

**END FOR**

**OUTPUT** firstIndexOfSmallestDiff , secondIndexOfSmallestDiff

arr[firstIndexOfSmallestDiff] , arr[secondIndexOfSmallestDiff]

**OUTPUT** firstIndexOfLargestDiff , secondIndexOfLargestDiff

arr[firstIndexOfLargestDiff] , arr[secondIndexOfLargestDiff]

1. **Motive behind design:** In a single loop, checking the absolute difference between current and next element, comparing them and storing the difference and indexes for future reference
2. **Time Complexity:** O(n)
3. **Maximum size of Stack growth:** O(1)