Assignment-6

Ans 1-

```
s = "IDID"
def diStringMatch(s: str) -> list[int]:
        ans = []
        low = 0
        high = len(s)
        for i in s:
            if i=="I":
                ans.append(low)
                low+=1
            else:
                ans.append(high)
                high-=1
        ans.append(high)
        return ans
diStringMatch(s)
```

Ans 2-

```
matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]]
target = 3
def searchMatrix(matrix: list[list[int]], target: int) ->
bool:
    rows = len(matrix)
    cols = len(matrix[0])
    row = 0
    for row in range(rows):
        if matrix[row][0] > target:
            row = row - 1
            break
    else:
        row = rows - 1
    for i in range(cols):
```

```
        if matrix[row][i] == target:
            return(True)
            break
    return(False)
searchMatrix(matrix , target)
```

Ans 3-

```
arr = [2,1]

def validMountainArray(arr: list[int]) -> bool:
    m = arr.index(max(arr))
    if m == 0 or m == len(arr) -1:
        print(False)
    else:
        flag = True
        i = 0
        while (i < m):
            if arr[i] >= arr[i+1]:
                flag = False
                break
            else:
                i+=1
        i = m+1
        if (flag):
            while i < len(arr):
                if arr[i-1] <= arr[i]:
                    return False
                    break
                else:
                    i+=1
        else:
            return False
        return True
validMountainArray(arr)
```

Ans 4-

```
arr = [0,1,0]
def findMaxLength(nums: list[int]) -> int:
    max_length =0
    hash={}
    count=0
    for i in range(len(nums)):
        current=nums[i]
        if current==0:
            count-=1 # decrement our count if our current
element is 0
        else:
            # increment our count if current element is 1
            count+=1

        if count==0:
            # if count is 0, we have a new subarray with
length+1
            max_length=i+1
        if count in hash:
            max_length=max(max_length,i-hash[count])
        else:
            hash[count]=i
    return max_length
findMaxLength(arr)
```

Ans 5-

```
nums1 = [5,3,4,2]
nums2 = [4,2,2,5]
def minProductSum(nums1:list[int], nums2:list[int])->int:
    """calculates and returns the minimum product of 2 arrays
    nums1 and nums2

    Args:
        nums1 (list[int]): array of numbers
        nums2 (list[int]): array of numbers
```

```
    Returns:
        int: minimum product sum of 2 arrays
    """
    nums1.sort()
    nums2.sort()
    n = len(nums1)

    result = 0
    for i in range(n):
        result += (nums1[i] * nums2[n-i-1])

    return(result)
minProductSum(nums1, nums2)
```

Ans 6-

```
changed = [1,3,4,2,6,8]
def findOriginalArray(changed: list[int]) -> list[int]:
    """checks if an array is doubled and returns the original
    array or empty list

    Args:
        changed (list[int]): doubled array

    Returns:
        list[int]: original array
    """
    from collections import Counter
    if len(changed)%2!=0:return []
    changed.sort()
    c=Counter(changed)
    ans=[]
    if c[0]%2==0:
        ans+=[0]*(c[0]//2)
```

```
    for i in c:
        if i==0 or c[i]==0:
            continue
        elif (i*2 not in c) or c[i]>c[i*2]:
            return []
        c[i*2]-=c[i]
        ans+=[i]*c[i]

    return ans
findOriginalArray(changed)
```

Ans 7-

```
n = 3
def generateMatrix(n: int) -> list[list[int]]:
        matrix = [[0]*n for _ in range(n)]
        r, c, dr, dc = 0, 0, 0, 1
        for current in range(1, n*n + 1):
            matrix[r][c] = current
            if not (0 <= r + dr < n and 0 <= c + dc < n and
matrix[r + dr][c + dc] == 0):
                dr, dc = dc, -dr
            r += dr
            c += dc
        return matrix
generateMatrix(3)
```

ANs 8-

```
mat1 = [[1,0,0],[-1,0,3]]
mat2 = [[7,0,0],[0,0,0],[0,0,1]]
def multiply(mat1: list[list[int]], mat2: list[list[int]]) ->
list[list[int]]:
  """sparse matrix multiplication of two matrices
  m X l and l x n
```

```
  Args:
      mat1 (list[list[int]]): sparse matrix m x l
      mat2 (list[list[int]]): sparse matrix l x n

  Returns:
      list[list[int]]: resulting matrix
  """
  m = len(mat1)
  n = len(mat2)
  l = len(mat2[0])
  ans = [[0] * l for _ in range(m)]
  nonZeroColIndicesInMat2 = [
      [j for j, a in enumerate(row) if a]
      for row in mat2
  ]

  for i in range(m):
    for j, a in enumerate(mat1[i]):
      if a == 0:
        continue
      # mat1's j-th column matches mat2's j-th row
      for colIndex in nonZeroColIndicesInMat2[j]:
        ans[i][colIndex] += a * mat2[j][colIndex]

  return ans
multiply(mat1, mat2)
```