Assignment-3

Ans1-

```
def threeSumClosest(nums, target):
    nums.sort()
    n = len(nums)
    closestSum = float('inf')

    for i in range(n - 2):
        left = i + 1
        right = n - 1

        while left < right:
            currentSum = nums[i] + nums[left] + nums[right]

            if abs(currentSum - target) < abs(closestSum -
target):
                closestSum = currentSum

            if currentSum > target:
                right -= 1
            elif currentSum < target:
                left += 1
            else:
                return target

    return closestSum
```

Ans 2-

```
def fourSum(nums, target):
    nums.sort()
    n = len(nums)
    quadruplets = []

    for a in range(n - 3):
        if a > 0 and nums[a] == nums[a - 1]:
            continue

        for b in range(a + 1, n - 2):
            if b > a + 1 and nums[b] == nums[b - 1]:
                continue

            left = b + 1
            right = n - 1

            while left < right:
                currentSum = nums[a] + nums[b] + nums[left] +
nums[right]

                if currentSum == target:
                    quadruplets.append([nums[a], nums[b],
nums[left], nums[right]])

                    while left < right and nums[left] ==
nums[left + 1]:
                        left += 1
                    while left < right and nums[right] ==
nums[right - 1]:
                        right -= 1

                    left += 1
                    right -= 1
                elif currentSum < target:
                    left += 1
                else:
                    right -= 1
```

```
    return quadruplets
```

Ans 3-

```
def nextPermutation(nums):
    n = len(nums)
    i = n - 2

    while i >= 0 and nums[i] >= nums[i + 1]:
        i -= 1

    if i >= 0:
        j = n - 1
        while j >= 0 and nums[j] <= nums[i]:
            j -= 1
        nums[i], nums[j] = nums[j], nums[i]

    left = i + 1
    right = n - 1
    while left < right:
        nums[left], nums[right] = nums[right], nums[left]
        left += 1
        right -= 1

    return nums
```

Ans 4-

```
def searchInsert(nums, target):
    left = 0
    right = len(nums) - 1
```

```
    while left <= right:
        mid = (left + right) // 2

        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return left
```

Ans-5

```
def plusOne(digits):
    n = len(digits)

    # Start from the rightmost digit
    for i in range(n - 1, -1, -1):
        digits[i] += 1

        if digits[i] == 10:
            digits[i] = 0
        else:
            # No carry left, return the digits array
            return digits

    # If there is a carry left, insert it at the beginning of
the array
    digits.insert(0, 1)
    return digits
```

Ans 6-

```python
def singleNumber(nums):
    result = 0

    for num in nums:
        result ^= num

    return result
```

Ans 7-

```python
def findMissingRanges(nums, lower, upper):
    result = []
    start = lower

    for num in nums:
        if num > start:
            result.append(getRange(start, num - 1))
        start = num + 1

    if start <= upper:
        result.append(getRange(start, upper))

    return result


def getRange(start, end):
    if start == end:
        return str(start)
    else:
        return str(start) + "->" + str(end)
```

Ans 8-

```python
def canAttendMeetings(intervals):
    intervals.sort(key=lambda x: x[0])

    for i in range(1, len(intervals)):
        if intervals[i][0] < intervals[i-1][1]:
            return False

    return True
```