ASSIGNMENT 3

1.

```
def threeSumClosest(nums, target):
    nums.sort()
    n = len(nums)
    closest_sum = float('inf')
    for i in range(n - 2):
        left = i + 1
        right = n - 1
        while left < right:
            current_sum = nums[i] + nums[left] + nums[right]
            if abs(current_sum - target) < abs(closest_sum - target):
                closest_sum = current_sum
            if current_sum < target:
                left += 1
            else:
                right -= 1
    return closest_sum
```

2.

```
def fourSum(nums, target):
    nums.sort()
    n = len(nums)
    result = []
    for i in range(n - 3):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        for j in range(i + 1, n - 2):
            if j > i + 1 and nums[j] == nums[j - 1]:
                continue
            left = j + 1
            right = n - 1
```

```
            while left < right:
                current_sum = nums[i] + nums[j] + nums[left] + nums[right]
                if current_sum == target:
                    result.append([nums[i], nums[j], 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 current_sum < target:
                    left += 1
                else:
                    right -= 1
    return result
```

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 > i 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
```

4.

```
def searchInsert(nums, target):
    left = 0
    right = len(nums)
    while left < right:
        mid = (left + right) // 2
        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid
    return left
```

5.

```
def plusOne(digits):
    n = len(digits)
    for i in range(n - 1, -1, -1):
        if digits[i] < 9:
            digits[i] += 1
            return digits
        else:
            digits[i] = 0
    return [1] + digits
```

6.

```
def singleNumber(nums):
    result = 0
    for num in nums:
        result ^= num
    return result
```

7.

```
def findMissingRanges(nums, lower, upper):
    def formatRange(lower, upper):
        if lower == upper:
            return str(lower)
        else:
            return str(lower) + "->" + str(upper)

    result = []
    prev = lower - 1
    for num in nums + [upper + 1]:
        if num == prev + 2:
            result.append(formatRange(prev + 1, num - 1))
        elif num > prev + 2:
            result.append(formatRange(prev + 1, num - 1))
        prev = num
    return result
```

8.

```
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
```