1)

```
def three_sum_closest(nums, target):
    nums.sort()  # Sort the array in ascending order
    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 current_sum == target:
        return current_sum

    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

# Test the function with the given example
nums = [-1, 2, 1, -4]
target = 1
result = three_sum_closest(nums, target)
print(result)
```

2)

```
def fourSum(nums, target):
    nums.sort()  # Sort the array to easily handle duplicates and improve efficiency
    result = []
    n = len(nums)

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

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

```
        c = b + 1
        d = n - 1

        while c < d:
            current_sum = nums[a] + nums[b] + nums[c] + nums[d]

            if current_sum == target:
            result.append([nums[a], nums[b], nums[c], nums[d]])

            # Skip duplicates
            while c < d and nums[c] == nums[c + 1]:
            c += 1
            while c < d and nums[d] == nums[d - 1]:
            d -= 1

            c += 1
            d -= 1

            elif current_sum < target:
            c += 1
            else:
            d -= 1

        return result


# Example usage
nums = [1, 0, -1, 0, -2, 2]
target = 0
result = fourSum(nums, target)
print(result)

3)
def nextPermutation(nums):
        # Find the first decreasing element from the right
        i = len(nums) - 2
        while i >= 0 and nums[i] >= nums[i + 1]:
        i -= 1

        if i >= 0:
        # Find the smallest element greater than nums[i] on the right side
        j = len(nums) - 1
        while nums[j] <= nums[i]:
        j -= 1
        # Swap nums[i] and nums[j]
        nums[i], nums[j] = nums[j], nums[i]
```

```
        # Reverse the subarray to the right of i
        left = i + 1
        right = len(nums) - 1
        while left < right:
        nums[left], nums[right] = nums[right], nums[left]
        left += 1
        right -= 1

# Example usage
nums = [1, 2, 3]
print("Input:", nums)
nextPermutation(nums)
print("Output:", nums)

4)
def searchInsert(nums, target):
        left, right = 0, 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

# Example usage
nums = [1, 3, 5, 6]
target = 5
print("Input:", nums)
print("Target:", target)
index = searchInsert(nums, target)
print("Output:", index)

5)
def plusOne(digits):
        n = len(digits)
        carry = 1

        for i in range(n - 1, -1, -1):
        digits[i] += carry

        if digits[i] < 10:
        carry = 0
```

```
            break

            digits[i] = 0

            if carry == 1:
            digits.insert(0, 1)

            return digits

# Example usage
digits = [1, 2, 3]
print("Input:", digits)
result = plusOne(digits)
print("Output:", result)

6)
def singleNumber(nums):
            result = 0
            for num in nums:
            result ^= num
            return result

# Example usage
nums = [2, 2, 1]
print("Input:", nums)
result = singleNumber(nums)
print("Output:", result)

7)
def findMissingRanges(nums, lower, upper):
            result = []
            prev = lower - 1

            for num in nums + [upper + 1]:
            if num - prev > 1:
            result.append(getRange(prev + 1, num - 1))
            prev = num

            return result

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

# Example usage
nums = [0, 1, 3, 50, 75]
```

```
lower = 0
upper = 99
print("Input:")
print("nums:", nums)
print("Range:", lower, "-", upper)
result = findMissingRanges(nums, lower, upper)
print("Output:", result)

8)
def canAttendMeetings(intervals):
        intervals.sort(key=lambda x: x[0])  # Sort the intervals based on the start time

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

        return True

# Example usage
intervals = [[0, 30], [5, 10], [15, 20]]
print("Input:", intervals)
result = canAttendMeetings(intervals)
print("Output:", result)
```