1)

```
def arrayPairSum(nums):
        nums.sort()  # Sort the array in ascending order
        max_sum = 0

        for i in range(0, len(nums), 2):
        max_sum += nums[i]  # Add the smaller element to max_sum

        return max_sum

# Example usage
nums = [1, 4, 3, 2]
result = arrayPairSum(nums)
print(result)  # Output: 4
```

2)

```
def max_candies(candyType):
    max_candies_count = len(candyType) // 2
    unique_candies = set(candyType)

    return min(max_candies_count, len(unique_candies))

# Example usage:
candyType = [1, 1, 2, 2, 3, 3]
max_diff_candies = max_candies(candyType)
print(max_diff_candies)
```

3)

```
def findLHS(nums):
    num_counts = {}
    max_length = 0

    # Count the frequency of each number in nums
    for num in nums:
        num_counts[num] = num_counts.get(num, 0) + 1

    # Check for harmonious subsequence
    for num in num_counts:
        if num + 1 in num_counts:
            max_length = max(max_length, num_counts[num] + num_counts[num + 1])

    return max_length

# Example usage:
nums = [1, 3, 2, 2, 5, 2, 3, 7]
max_length = findLHS(nums)
print(max_length)
```

4)

```
def canPlaceFlowers(flowerbed, n):
    count = 0
    i = 0
    size = len(flowerbed)

    while i < size:
        if flowerbed[i] == 0 and (i == 0 or flowerbed[i - 1] == 0) and (i == size - 1 or flowerbed[i +
1] == 0):
            flowerbed[i] = 1
            count += 1
            i += 1

        if count >= n:
            return True

        i += 1

    return False

# Example usage:
flowerbed = [1, 0, 0, 0, 1]
n = 1
can_plant = canPlaceFlowers(flowerbed, n)
print(can_plant)
```

5)

```
def maximum_product(nums):
        # Sort the array in ascending order
        nums.sort()

        # Get the length of the array
        n = len(nums)

        # Return the maximum of two possible products:
        # 1. Product of the three largest numbers
        # 2. Product of the two smallest numbers and the largest number
        return max(nums[n-1] * nums[n-2] * nums[n-3], nums[0] * nums[1] * nums[n-1])

# Example usage
nums = [1, 2, 3]
max_product = maximum_product(nums)
print(max_product)
```

6)

```
def search(nums, target):
        left = 0
```

```
        right = len(nums) - 1

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

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

        return -1

# Example usage
nums = [-1, 0, 3, 5, 9, 12]
target = 9
index = search(nums, target)
print(index)

7)
def minDifference(nums, k):
    n = len(nums)

    if n <= 1:
        return 0

    nums.sort()

    # Initialize the minimum score
    min_score = nums[-1] - nums[0]

    # Iterate through possible values for the first and last elements
    for i in range(k+1):
        # Calculate the current score
        current_score = nums[-1-i] - nums[i]

        # Update the minimum score
        min_score = min(min_score, current_score)

    return min_score

# Example usage
nums = [1]
k = 0
min_score = minDifference(nums, k)
print(min_score)
```

8)

```
def minDifference(nums, k):
    n = len(nums)

    if n <= 1:
        return 0

    nums.sort()

    # Initialize the minimum score
    min_score = nums[-1] - nums[0]

    # Iterate through possible values for the first and last elements
    for i in range(k+1):
        # Calculate the current score
        current_score = nums[-1-i] - nums[i]

        # Update the minimum score
        min_score = min(min_score, current_score)

    return min_score

# Example usage
nums = [1]
k = 0
min_score = minDifference(nums, k)
print(min_score)
```