**Assignment day – 1**

**By: Kunal Sudhir Mishra**

**Question-1**

**Find a pair with the given sum in an array**

**Given an unsorted integer array, find a pair with the given sum in it.**

**For example**

**Input: nums = [8, 7, 2, 5, 3, 1]target = 10 Output: Pair found (8, 2)orPair found (7, 3)**

**Code:**

```
def find_pair(nums, target):
    n = len(nums)

    for i in range(n - 1):
        for j in range(i + 1, n):
            if nums[i] + nums[j] == target:
                print(f"Pair found ({nums[i]}, {nums[j]})")
                return True

    print("No pair found with the given sum")
    return False
```

```
nums = [8, 7, 2, 5, 3, 1]

target = 10

find_pair_with_sum_bruteforce(nums, target)
```

**Question-2**

**Given an integer array, replace each element with the product of every other element without using the division operator.**

**For example,**

**Input: { 1, 2, 3, 4, 5 }Output: { 120, 60, 40, 30, 24 } Input: { 5, 3, 4, 2, 6, 8 }Output: { 1152, 1920, 1440, 2880, 960, 720 }**

**Code :**

```
def replace_with_product(nums):
    n = len(nums)



    left_products = [1] * n
    right_products = [1] * n



    left_product = 1
    for i in range(1, n):
        left_product *= nums[i - 1]
        left_products[i] = left_product
```

```
    right_product = 1
    for i in range(n - 2, -1, -1):
        right_product *= nums[i + 1]
        right_products[i] = right_product

    result = [left_products[i] * right_products[i] for i in range(n)]

    return result

nums = [1, 2, 3, 4, 5]
result = replace_with_product(nums)
print(result)
```

**Question-3**

**Maximum Sum Circular Subarray**

**Given a circular integer array, find a subarray with the largest sum in it.**

**For example :Input: {2, 1, -5, 4, -3, 1, -3, 4, -1} Output: Subarray with the largest sum is {4, -1, 2, 1} with sum 6.**

**Code :**

```
def max_subarray_sum_circular(nums):
    n = len(nums)
    max_sum_without_wrap = float('-inf')
    current_sum = 0
    for num in nums:
        current_sum = max(num, current_sum + num)
        max_sum_without_wrap = max(max_sum_without_wrap, current_sum)
    total_sum = sum(nums)
    min_sum_subarray = float('inf')
    current_sum = 0
    for num in nums:
        current_sum = min(num, current_sum + num)
        min_sum_subarray = min(min_sum_subarray, current_sum)
```

```
    max_sum_with_wrap = total_sum - min_sum_subarray
    if max(nums) < 0:
        return max_sum_without_wrap
    return max(max_sum_without_wrap, max_sum_with_wrap)
nums = [2, 1, -5, 4, -3, 1, -3, 4, -1]
result = max_subarray_sum_circular(nums)
print(result)
```

**Question-4:**

**Find the maximum difference between two array elements that satisfies the given constraints**

**Given an integer array, find the maximum difference between two elements in it such that the smaller element appears before the larger element.**

**For example:Input: { 2, 7, 9, 5, 1, 3, 5 } Output: The maximum difference is 7. The pair is (2, 9)**

**Code:**

```
def max_difference(nums):
    if len(nums) < 2:
        return "Array should contain at least two elements."
    min_element = nums[0]
    max_difference = nums[1] - nums[0]
```

```
    pair = (min_element, nums[1])
    for num in nums[1:]:
        if num - min_element > max_difference:
            max_difference = num - min_element
            pair = (min_element, num)
        if num < min_element:
            min_element = num
    return max_difference, pair


nums = [2, 7, 9, 5, 1, 3, 5]
result, pair = max_difference(nums)
print(f"The maximum difference is {result}. The pair is {pair}.")
```

**Question:5**

**Given an array of integers of size N, the task is to find the first non-repeating element in this array.**

**Examples:**

**Input: {-1, 2, -1, 3, 0}**

**Output: 2**

**Explanation: The first number that does not repeat is : 2**

**Input: {9, 4, 9, 6, 7, 4}**

**Output: 6**

**Code:**

```
def first_non_repeating_element(nums):
    # Dictionary to store the frequency of each element
    frequency = {}

    # List to maintain the order of non-repeating elements
    non_repeating_order = []

    for num in nums:
        # Update the frequency
        frequency[num] = frequency.get(num, 0) + 1
```

```
        # If the element is non-repeating, add it to the order list
        if frequency[num] == 1:
            non_repeating_order.append(num)
        else:
            # If the element is repeating, remove it from the order list
            if num in non_repeating_order:
                non_repeating_order.remove(num)

    # If there are non-repeating elements, return the first one; otherwise, return None
    return non_repeating_order[0] if non_repeating_order else None

# Example usage
nums = [9, 4, 9, 6, 7, 4]
result = first_non_repeating_element(nums)

if result is not None:
    print(f"The first non-repeating element is {result}.")
else:
    print("No non-repeating element found.")
```

**Question:6**

**Minimize the maximum difference between the heights**

**Given the heights of N towers and a value of K, Either increase or decrease the height of every tower by K (only once) where K > 0. After modifications, the task is to minimize the difference between the heights of the longest and the shortest tower and output its difference.**

**Examples:**

**Input: arr[] = {1, 15, 10}, k = 6**

**Output: Maximum difference is 5.**

**Explanation: Change 1 to 7, 15 to 9 and 10 to 4. Maximum difference is 5 (between 4 and 9). We can't get a lower difference.**

**Input: arr[] = {1, 5, 15, 10}, k = 3**

**Output: Maximum difference is 8, arr[] = {4, 8, 12, 7}**

**Code :**

```
def minimize_max_difference(arr, k):
    n = len(arr)

    # Sort the array
    arr.sort()

    # Initialize the new heights after modification
```

```
modified_heights = []

# Initialize the new maximum and minimum heights
new_max_height = 0
new_min_height = 0

# Iterate through each tower
for i in range(n):
    if arr[i] - k >= arr[0] + k:
        # Subtract K from the current tower
        modified_heights.append(arr[i] - k)
    else:
        # Add K to the current tower
        modified_heights.append(arr[0] + k)

    new_max_height = max(new_max_height, modified_heights[i])
    new_min_height = min(modified_heights[i], modified_heights[0])

# Calculate the new difference
new_difference = new_max_height - new_min_height

return new_difference
```

```
# Example usage
arr1 = [1, 15, 10]
k1 = 6
result1 = minimize_max_difference(arr1, k1)
print(f"Maximum difference is {result1}.")

arr2 = [1, 5, 15, 10]
k2 = 3
result2 = minimize_max_difference(arr2, k2)
print(f"Maximum difference is {result2}.")
```