ASSIGNMENT 1

Name: Shivam Singh

1.

```python
def find_pairs_with_sum(nums, target):
    pairs = []
    num_dict = {}

    for num in nums:

        diff = target - num

        if diff in num_dict:

            pairs.append((num, diff))

        num_dict[num] = True

    return pairs if pairs else "No pairs found"

# Example
nums = [8, 7, 2, 5, 3, 1]
target = 10
result = find_pairs_with_sum(nums, target)
if result != "No pairs found":
    for pair in result:
        print(f"Pair found {pair}")
else:
    print(result)
```

2.

```python
def product_except_self(nums):
    n = len(nums)
    left_products = [1] * n
    right_products = [1] * n
    result = [1] * n

    # the left
    for i in range(1, n):
        left_products[i] = left_products[i - 1] * nums[i - 1]

    # the right
    for i in range(n - 2, -1, -1):
        right_products[i] = right_products[i + 1] * nums[i + 1]

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

    return result

# Example
input1 = [1, 2, 3, 4, 5]
output1 = product_except_self(input1)
print(output1)
```

3.

```python
def k(nums):
    max_sum = float('-inf')
    current_sum = 0

    for num in nums:
        current_sum = max(num, current_sum + num)
        max_sum = max(max_sum, current_sum)

    return max_sum

def max_subarray_sum_circular(nums):

    max_k = k(nums)

    total_sum = sum(nums)
    inverted_nums = [-x for x in nums]  # Inverting the array
    max_wrap = total_sum + k(inverted_nums[1:-1])

    # Take the maximum of the two cases
    return max(max_k, max_wrap)
```

```
# Example
arr = [2, 1, -5, 4, -3, 1, -3, 4, -1]
result = max_subarray_sum_circular(arr)
print(f"Subarray with the largest sum is {result}")
```

4.

```
def max_difference_with_pair(nums):
    if len(nums) < 2:
        return "Array should have at least two elements"

    min_element = nums[0]
    max_diff = nums[1] - nums[0]
    pair = (nums[0], nums[1])

    for i in range(1, len(nums)):
        if nums[i] - min_element > max_diff:
            max_diff = nums[i] - min_element
            pair = (min_element, nums[i])

        if nums[i] < min_element:
            min_element = nums[i]

    return max_diff, pair

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

5.

```python
def first_non_repeating_element(nums):
    count = {}

    # Count the frequency of each element
    for num in nums:
        if num in count:
            count[num] += 1
        else:
            count[num] = 1

    # Find the first element with count 1
    for num in nums:
        if count[num] == 1:
            return num

    return None  # If no non-repeating element found

# Example usage

arr1 = [-1, 2, -1, 3, 0]
result1 = first_non_repeating_element(arr1)
print("Output:", result1)
```

6.

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

    # Sort the array to easily determine the tallest and shortest towers
    arr.sort()

    # Initialize result with the difference between the tallest and shortest
towers
    result = arr[n - 1] - arr[0]

    # Consider the maximum and minimum heights after adding/subtracting K from
each tower
    max_height = arr[n - 1] - k
    min_height = arr[0] + k

    # Handle edge cases where swapping might yield better results
    if min_height > max_height:
        min_height, max_height = max_height, min_height

    # Update result by comparing with the differences obtained after possible
modifications
    for i in range(1, n - 1):
        add = arr[i] + k
        subtract = arr[i] - k

        # Check if the modification leads to a better result
        if subtract >= min_height or add <= max_height:
            continue

        # Update min_height or max_height accordingly
        if max_height - subtract <= add - min_height:
            min_height = subtract
        else:
            max_height = add

    # Update the final result by considering the minimized difference
    return min(result, max_height - min_height)

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