Experiment 9

Student Name: Rhythm Tyagi **UID:** 22BCS17203
Branch: CSE **Section/Group:**IOT_NTPP_602-A
Semester: 6th **Date of Performance:**03-05-25
Subject Name: AP2 **Subject Code:** 22CSP-351

Aim:

1) Two Sum: Given an array of integers, return indices of the two numbers such that they add up to a specific target.

2) Palindrome Number: Determine hether an integer is a palindrome.

3) Maximum Subarray: Find the contiguous subarray (containing at least one number) that has the largest sum and return its sum.

Objective: Develop efficient algorithms to solve fundamental array and number problems, including finding two numbers summing to a target, checking integer palindromes, and identifying the maximum subarray sum.

Algorithm 1:

1. Initialize a HashMap to store numbers and their indices.
2. Iterate through the array using a loop:
3. Compute the complement (target - current_number).
4. Check if the complement exists in the HashMap.
5. If found, return the indices of the complement and the current number.
6. Otherwise, store the current number and its index in the HashMap.
7. Return an empty array if no solution is found

Code 1:

```
import java.util.*;

public class TwoSum {

    public static int[] findTwoSum(int[] nums, int target) {
```

```
        Map<Integer, Integer> map = new HashMap<>();
            for (int i = 0; i < nums.length; i++) {
        int complement = target - nums[i];
        if (map.containsKey(complement)) {
            return new int[]{map.get(complement), i};
        }
        map.put(nums[i], i);
    }
    return new int[]{}; // Return empty if no solution
  }
  public static void main(String[] args) {
    int[] nums = {2, 7, 11, 15};
    int target = 9;
    int[] result = findTwoSum(nums, target);
            System.out.println("Indices: " + Arrays.toString(result));
  }
}
```

Output 1:

```
PS R:\Java 51> java TwoSum
>>
Indices: [0, 1]
```

Algorithm 2:

1. Handle edge cases:
2. If x is negative or ends with 0 (except 0 itself), return false.
3. Reverse the number:
4. Initialize reversed = 0.
5. Extract digits one by one and build the reversed number.
6. Compare original and reversed numbers:

7. If they are equal, return true (palindrome).
8. Otherwise, return false

Code 2:

```
public class PalindromeNumber {
   public static boolean isPalindrome(int x) {
      if (x < 0 || (x % 10 == 0 && x != 0)) return false; // Negative & trailing zero cases
      int reversed = 0, original = x;
      while (x > 0) {
         reversed = reversed * 10 + x % 10;
         x /= 10;
      }
      return original == reversed;
   }
  public static void main(String[] args) {
      int num = 121;
      System.out.println(num + " is Palindrome? " + isPalindrome(num));
   }
}
```

Output 2:

```
121 is Palindrome? true
```

Algorithm 3:

1. Initialize variables:
2. maxSum = nums[0] (stores the maximum sum found).

3. currentSum = nums[0] (tracks the current subarray sum).
4. Iterate through the array (starting from index 1):
5. Update currentSum as the maximum of the current element or currentSum + current element.
6. Update maxSum if currentSum is greater.
7. Return maxSum as the maximum subarray sum

Code 3:

```
public class MaximumSubarray {
   public static int maxSubArray(int[] nums) {
      int maxSum = nums[0], currentSum = nums[0];
      for (int i = 1; i < nums.length; i++) {
         currentSum = Math.max(nums[i], currentSum + nums[i]);
         maxSum = Math.max(maxSum, currentSum);
      }
      return maxSum;
   }
   public static void main(String[] args) {
      int[] nums = {-2, 1, -3, 4, -1, 2, 1, -5, 4};
      System.out.println("Maximum Subarray Sum: " + maxSubArray(nums));
   }
}
```

Output:

```
Maximum Subarray Sum: 6
```

Learning Outcomes:

1. Understand HashMap-based approach for Two Sum.
2. Implement integer reversal for palindrome detection.
3. Apply Kadane's algorithm for maximum subarray sum.
4. Optimize solutions for time and space complexity.
5. Strengthen problem-solving skills in array manipulation