Day -1 Practical Work

Mridul Bhardwaj

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) or Pair found (7, 3)

Solution :

```
#include <stdio.h>

void pair(int a[], int n, int s)

{

   for (int i = 0; i < n - 1; i++)

   {

      for (int j = i + 1; j < n; j++)

      {

         if (a[i] + a[j] == s)

         {

            printf("Pair found (%d, %d)\n", a[i], a[j]);
```

```
            return;
          }
        }
    }
    printf("Pair not found");
}
int main()
{
  int a[] = { 8, 9, 2, 3, 6, 1};
    int s = 11;
    int n = sizeof(a)/sizeof(a[0]);
    pair(a, n, s);
    return 0;
}
```

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 }

Solution :

```
#include <stdio.h>
```

```
void prod(int a[], int n)
{    if (n == 0) {
        return;
    }
    int l[n], r[n];
    l[0] = 1;
    for (int i = 1; i < n; i++) {
        l[i] = a[i - 1] * l[i - 1];
    }
     r[n - 1] = 1;
    for (int j = n - 2; j >= 0; j--) {
        r[j] = a[j + 1] * r[j + 1];
    }
    for (int i = 0; i < n; i++) {
        a[i] = l[i] * r[i];
    }
}
int main() {
    int a[] = { 1, 3, 2, 4, 6, 5 };
    int n = sizeof(a) / sizeof(a[0]);
    prod(a, n);
    for (int i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    return 0;
```

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.

Solution :

```
import java.util.Scanner;
public class MaximumSumCircularSubarray {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter the size");
        int n  = sc.nextInt();
        int[] arr= new int[n];
        System.out.println("enter the elements");
        for(int i=0;i<n;i++)
            arr[i] = sc.nextInt();

        int maxNormal = 0;
        int maxCircular = 0;
```

```
        for (int i = 0; i < n; i++) {
            maxNormal = Math.max(arr[i], maxNormal + arr[i]);
            maxCircular += arr[i];
            arr[i] = -arr[i];
        }

        maxCircular = maxCircular + maxSubarraySum(arr);

        int result = Math.max(maxNormal, maxCircular);

        System.out.println("Subarray with the largest sum is: " + result);
    }

    private static int maxSubarraySum(int[] nums) {
        int maxSum = nums[0];
        int 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;
    }
}
```

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)

Solution :

```
#include <iostream>

int maxDiff(int a[], int n)

  {

    int max = a[1] - a[0];

    int i, j;

    for (i = 0; i < n; i++)

    {

      for (j = i + 1; j < n; j++)

      {

        if (a[j] - a[i] > max)

          max = a[j] - a[i];

      }

    }
```

```
        return max;
    }
int main()
{
    int arr[] = {1, 2, 9, 10, 11};
  int n = sizeof(arr) / sizeof(arr[0]);
  cout << "Maximum difference is " << maxDiff(arr, n);
    return 0;
}
```

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

Solution :

```
#include <iostream>
int firstNonRepeating(int arr[], int n)
```

```
{
    for (int i = 0; i < n; i++) {
        int j;
        for (j = 0; j < n; j++)
            if (i != j && arr[i] == arr[j])
                break;
        if (j == n)
            return arr[i];
    }
    return -1;
}
int main()
{
    int arr[] = {-1, 2, -1, 3, 0};
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << firstNonRepeating(arr, n);

    return 0;
}
```

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}

Solution :

```
import java.util.Arrays;

import java.util.Scanner;

public class MinDiff {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      System.out.println("enter the size");

      int n  = sc.nextInt();

      System.out.println("enter k");

      int k= sc.nextInt();
```

```
        int[] arr= new int[n];
        int avg=0;
        System.out.println("enter elements");
        for(int i=0;i<n;i++){
           arr[i] = sc.nextInt();
           avg+=arr[i];
        }
        avg = avg/n;
        for(int i=0;i<n;i++){
           if(arr[i]>avg)
              arr[i]-=k;
           else
              arr[i]+=k;
        }
        Arrays.sort(arr);
        int diff = arr[n-1]-arr[0];
        System.out.println("the minimum difference is "+ diff);
    }
}
```