Problem Solving C++ Lab
Assignment -3

Q1]

```
#include <iostream>
#include <vector>
#include <unordered_map>

bool canArrange(std::vector<int>& arr, int k) {
    std::unordered_map<int, int> freq;

    for (int num : arr) {
        int remainder = (num % k + k) % k;
        freq[remainder]++;
    }

    if (freq[0] % 2 != 0)
        return false;

    for (int i = 1; i <= k / 2; i++) {
        if (freq[i] != freq[k - i])
            return false;
    }

    return true;
}

int main() {
    std::vector<int> arr = {1, 2, 3, 4, 5, 10, 6, 7, 8, 9};
    int k = 5;

    if (canArrange(arr, k)) {
```

```
        std::cout << "It is possible to arrange the array into pairs with
sum divisible by " << k << ".\n";
    } else {
        std::cout << "It is not possible to arrange the array into pairs with
sum divisible by " << k << ".\n";
    }

    return 0;
}
```

Q2]

```
#include <iostream>
#include <vector>

int maxCircularSum(std::vector<int>& arr) {
    int n = arr.size();

    int maxStraightSum = INT_MIN, minStraightSum = INT_MAX;
    int arraySum = 0, minSum = 0, maxSum = 0;

    for (int i = 0; i < n; i++) {
        arraySum += arr[i];

        maxSum += arr[i];
        if (maxSum < arr[i]) {
            maxSum = arr[i];
        }

        minSum += arr[i];
        if (minSum > arr[i]) {
            minSum = arr[i];
        }
```

```
        maxStraightSum = std::max(maxStraightSum, maxSum);
        minStraightSum = std::min(minStraightSum, minSum);
    }

    if (arraySum == minStraightSum) {
        return maxStraightSum;
    } else {
        return std::max(maxStraightSum, arraySum - minStraightSum);
    }
}

int main() {
    std::vector<int> arr = {8, -4, 3, -5, 4};

    std::cout << "The maximum circular sum is: " <<
maxCircularSum(arr) << std::endl;

    return 0;
}
```

Q3]

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <unordered_map>

int minSwaps(std::vector<int>& arr) {
    int n = arr.size();
    std::vector<int> temp = arr;

    std::sort(temp.begin(), temp.end());
```

```
    std::unordered_map<int, int> indexMap;
    for (int i = 0; i < n; i++) {
        indexMap[temp[i]] = i;
    }

    int swaps = 0;
    for (int i = 0; i < n; i++) {
        if (arr[i] != temp[i]) {
            swaps++;
            int correctValue = temp[i];
            std::swap(arr[i], arr[indexMap[correctValue]]);
        }
    }

    return swaps;
}

int main() {
    std::vector<int> arr1 = {4, 3, 2, 1};
    std::vector<int> arr2 = {1, 5, 4, 3, 2};

    std::cout << "Minimum swaps for arr1: " << minSwaps(arr1) <<
std::endl;
    std::cout << "Minimum swaps for arr2: " << minSwaps(arr2) <<
std::endl;

    return 0;
}
```

Q4]

```
#include <iostream>
```

```
#include <vector>

void rotateMatrix(std::vector<std::vector<int>>& matrix) {
    int n = matrix.size();

    for (int i = 0; i < n / 2; i++) {
        for (int j = i; j < n - i - 1; j++) {
            int temp = matrix[i][j];
            matrix[i][j] = matrix[j][n - i - 1];
            matrix[j][n - i - 1] = matrix[n - i - 1][n - j - 1];
            matrix[n - i - 1][n - j - 1] = matrix[n - j - 1][i];
            matrix[n - j - 1][i] = temp;
        }
    }
}

void printMatrix(const std::vector<std::vector<int>>& matrix) {
    for (const auto& row : matrix) {
        for (int num : row) {
            std::cout << num << " ";
        }
        std::cout << std::endl;
    }
}

int main() {
    std::vector<std::vector<int>> matrix = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
```

```
    std::cout << "Original Matrix:\n";
    printMatrix(matrix);

    rotateMatrix(matrix);

    std::cout << "\nMatrix after rotation by 90 degrees in anti-clockwise
direction:\n";
    printMatrix(matrix);

    return 0;
}
```

Q5]

```
#include <iostream>
#include <vector>
#include <deque>

std::vector<int> maxSlidingWindow(std::vector<int>& nums, int k) {
    std::vector<int> result;
    std::deque<int> dq;

    for (int i = 0; i < nums.size(); i++) {
        while (!dq.empty() && dq.front() < i - k + 1) {
            dq.pop_front();
        }

        while (!dq.empty() && nums[dq.back()] < nums[i]) {
            dq.pop_back();
        }

        dq.push_back(i);
```

```
        if (i >= k - 1) {
            result.push_back(nums[dq.front()]);
        }
    }

    return result;
}

int main() {
    std::vector<int> nums = {1, 3, -1, -3, 5, 3, 6, 7};
    int k = 3;

    std::vector<int> result = maxSlidingWindow(nums, k);
```

Q6]

```
#include <iostream>
#include <vector>
#include <deque>

std::vector<int> maxSlidingWindow(std::vector<int>& nums, int k) {
    std::vector<int> result;
    std::deque<int> dq;

    for (int i = 0; i < nums.size(); i++) {
        while (!dq.empty() && dq.front() < i - k + 1) {
            dq.pop_front();
        }

        while (!dq.empty() && nums[dq.back()] < nums[i]) {
            dq.pop_back();
        }
```

```
        dq.push_back(i);

        if (i >= k - 1) {
            result.push_back(nums[dq.front()]);
        }
    }

    return result;
}

int main() {
    std::vector<int> nums = {1, 3, -1, -3, 5, 3, 6, 7};
    int k = 3;

    std::vector<int> result = maxSlidingWindow(nums, k);

    std::cout << "Maximum number in sliding windows of size " << k <<
":\n";
    for (int num : result) {
        std::cout << num << " ";
    }
    std::cout << std::endl;

    return 0;
}
```