Name: Keshav Kumar Soni

UID:22BCS12587

Section: KPIT_901/B

DAY 5

1. Minimum Number of Moves to Seat Everyone

Question:
You are given n available seats and n students standing in a room.

- seats[i] is the position of the i-th seat.
- students[j] is the position of the j-th student.

You can perform the following move any number of times:

- Increase or decrease the position of a student by 1.

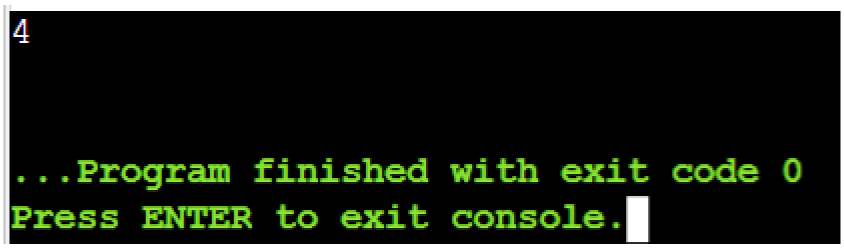
Return the minimum number of moves required to move each student to a seat such that no two students are in the same seat.

Code:

```
#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;


int minMovesToSeat(vector<int>& seats, vector<int>& students) {

   sort(seats.begin(), seats.end());

   sort(students.begin(), students.end());

   int moves = 0;

   for (int i = 0; i < seats.size(); ++i) {

      moves += abs(seats[i] - students[i]);

   }

   return moves;

}
```

```
int main() {
    vector<int> seats = {3, 1, 5};
    vector<int> students = {2, 7, 4};
    cout << minMovesToSeat(seats, students) << endl;
    return 0;
}
```

Output:



```
4


...Program finished with exit code 0
Press ENTER to exit console.
```

2. Squares of a Sorted Array

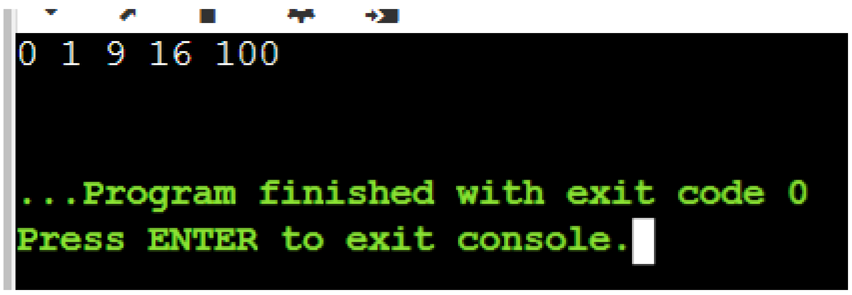
Question:
Given an integer array nums sorted in non-decreasing order, return an array of the squares of each number sorted in non-decreasing order.

Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;


vector<int> sortedSquares(vector<int>& nums) {
    for (int i = 0; i < nums.size(); ++i) {
        nums[i] = nums[i] * nums[i];
    }
    sort(nums.begin(), nums.end());
    return nums;
}
```

```
int main() {
    vector<int> nums = {-4, -1, 0, 3, 10};
    vector<int> result = sortedSquares(nums);
    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;
    return 0;
}
```

Output:



```
0 1 9 16 100


...Program finished with exit code 0
Press ENTER to exit console.
```

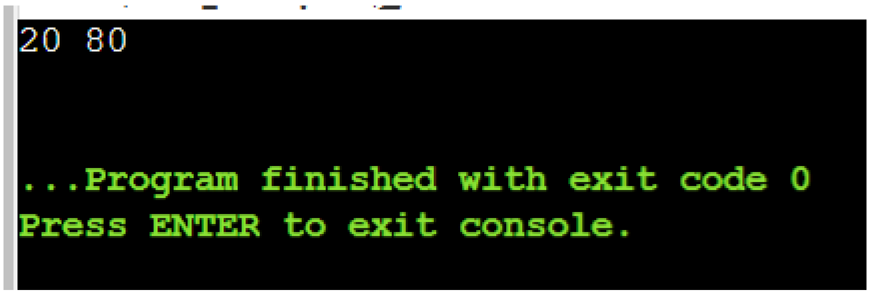
3. Common Elements in Three Sorted Arrays

Question:
Given three sorted arrays, find the elements that are common in all three arrays. If no common elements exist, return -1.

Code:

```
#include <iostream>
#include <vector>
using namespace std;

vector<int> commonElements(vector<int>& arr1, vector<int>& arr2, vector<int>& arr3) {
    vector<int> result;
    int i = 0, j = 0, k = 0;
    while (i < arr1.size() && j < arr2.size() && k < arr3.size()) {
        if (arr1[i] == arr2[j] && arr2[j] == arr3[k]) {
            if (result.empty() || result.back() != arr1[i]) {
                result.push_back(arr1[i]);
            }
            i++; j++; k++;
        } else if (arr1[i] < arr2[j]) {
            i++;
        } else if (arr2[j] < arr3[k]) {
            j++;
        } else {
            k++;
        }
    }
    return result.empty() ? vector<int>{-1} : result;
}

int main() {
    vector<int> arr1 = {1, 5, 10, 20, 40, 80};
    vector<int> arr2 = {6, 7, 20, 80, 100};
    vector<int> arr3 = {3, 4, 15, 20, 30, 70, 80, 120};
    vector<int> result = commonElements(arr1, arr2, arr3);
    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;
    return 0;
}
```

Output:



```
20 80


...Program finished with exit code 0
Press ENTER to exit console.
```

4. Sort Even and Odd Indices Independently

Question:
You are given a 0-indexed integer array nums. Rearrange the values of nums according to the following rules:

1. Sort the values at odd indices of nums in non-increasing order.
2. Sort the values at even indices of nums in non-decreasing order.

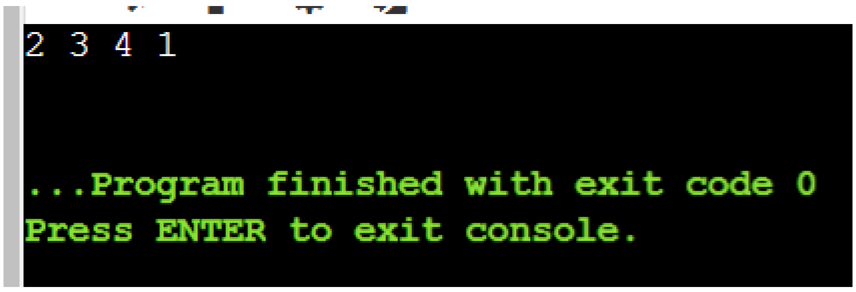
Return the array formed after rearranging the values of nums.

Code:

```
#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;


vector<int> sortEvenOdd(vector<int>& nums) {

  vector<int> even, odd;

  for (int i = 0; i < nums.size(); i++) {

    if (i % 2 == 0) {

      even.push_back(nums[i]);

    } else {

      odd.push_back(nums[i]);

    }

  }


  sort(even.begin(), even.end());
```

```
    sort(odd.rbegin(), odd.rend());

    for (int i = 0, j = 0, k = 0; i < nums.size(); i++) {
        if (i % 2 == 0) {
            nums[i] = even[j++];
        } else {
            nums[i] = odd[k++];
        }
    }

    return nums;
}

int main() {
    vector<int> nums = {4, 1, 2, 3};
    vector<int> result = sortEvenOdd(nums);
    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;
    return 0;
}
```

Output:



```
2 3 4 1


...Program finished with exit code 0
Press ENTER to exit console.
```

5. Leftmost and Rightmost Index

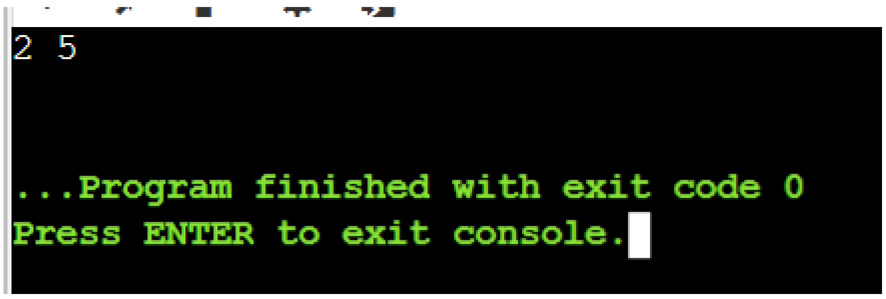
Question:
Given a sorted array with possibly duplicate elements, find the indexes of the first and last occurrences of an element X in the given array.

If the element is not present in the array, return {-1, -1} as a pair.

Code:

```
#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;


vector<int> findFirstAndLast(vector<int>& nums, int X) {

    int left = -1, right = -1;

    int low = 0, high = nums.size() - 1;


    // Find the first occurrence

    while (low <= high) {

        int mid = low + (high - low) / 2;

        if (nums[mid] == X) {

            left = mid;

            high = mid - 1;

        } else if (nums[mid] < X) {

            low = mid + 1;

        } else {

            high = mid - 1;

        }

    }


    low = 0, high = nums.size() - 1;
    // Find the last occurrence
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (nums[mid] == X) {
            right = mid;
            low = mid + 1;
        } else if (nums[mid] < X) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    return {left, right};
}

int main() {
    vector<int> nums = {1, 3, 5, 5, 5, 5, 67, 123, 125};
    int X = 5;
    vector<int> result = findFirstAndLast(nums, X);
    cout << result[0] << " " << result[1] << endl;
    return 0;
}
```

Output:



```
2 5


...Program finished with exit code 0
Press ENTER to exit console.
```