DAY-5 WWC

NAME: Siddharth Lal

UID: 22BCS13410

Problem-1(VeryEasy)

CODE:

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

int findFirstOccurrence(int k, const vector<int>& arr) {
    for (int i = 0; i < arr.size(); ++i) {
        if (arr[i] == k) {
            return i + 1;
        }
    }
    return -1;
}

int main() {
    int k1 = 16;
    vector<int> arr1 = {9, 7, 16, 16, 4};
    cout << findFirstOccurrence(k1, arr1) << endl;

    int k2 = 98;
    vector<int> arr2 = {1, 22, 57, 47, 34, 18, 66};
    cout << findFirstOccurrence(k2, arr2) << endl;
    return 0;
}
```

OUTPUT:

```
3
-1


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

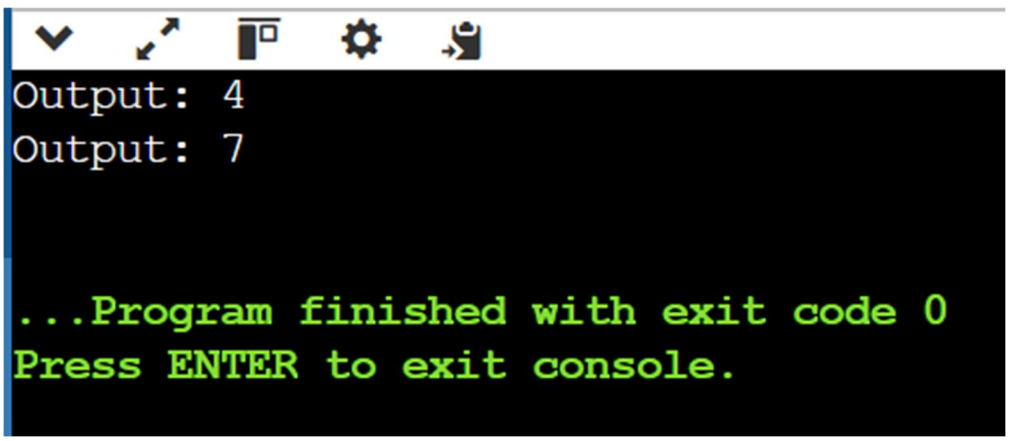
Problem-2(Easy)

CODE:

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

int minMovesToSeat(vector<int>& seats, vector<int>& students) {
    sort(seats.begin(), seats.end());
    sort(students.begin(), students.end());

    int totalMoves = 0;
    for (int i = 0; i < seats.size(); ++i) {
        totalMoves += abs(seats[i] - students[i]);
    }
    return totalMoves;
}

int main() {
```

```
    vector<int> seats1 = {3, 1, 5};
    vector<int> students1 = {2, 7, 4};
    cout << "Output: " << minMovesToSeat(seats1, students1) << endl;

    vector<int> seats2 = {4, 1, 5, 9};
    vector<int> students2 = {1, 3, 2, 6};
    cout << "Output: " << minMovesToSeat(seats2, students2) << endl;

    return 0;
}
```

OUTPUT:



Problem-3(Medium)

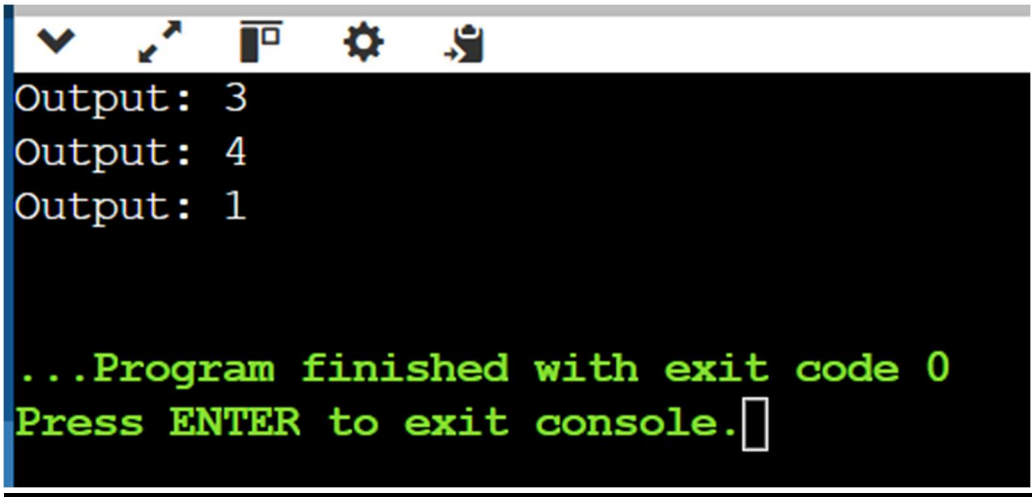
CODE:

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

int findSmallestMissingPositive(vector<int>& arr) {
    int n = arr.size();

    int j = 0;
```

```
    for (int i = 0; i < n; ++i) {
        if (arr[i] > 0) {
            swap(arr[i], arr[j]);
            ++j;
        }
    }

    for (int i = 0; i < j; ++i) {
        int val = abs(arr[i]);
        if (val - 1 < j && arr[val - 1] > 0) {
            arr[val - 1] = -arr[val - 1];
        }
    }

    for (int i = 0; i < j; ++i) {
        if (arr[i] > 0) {
            return i + 1;
        }
    }

    return j + 1;
}

int main() {
    vector<int> arr1 = {2, -3, 4, 1, 1, 7};
    cout << "Output: " << findSmallestMissingPositive(arr1) << endl;

    vector<int> arr2 = {5, 3, 2, 5, 1};
    cout << "Output: " << findSmallestMissingPositive(arr2) << endl;

    vector<int> arr3 = {-8, 0, -1, -4, -3};
    cout << "Output: " << findSmallestMissingPositive(arr3) << endl;
```

```
    return 0;
}
```

OUTPUT:



Problem-4(Hard)

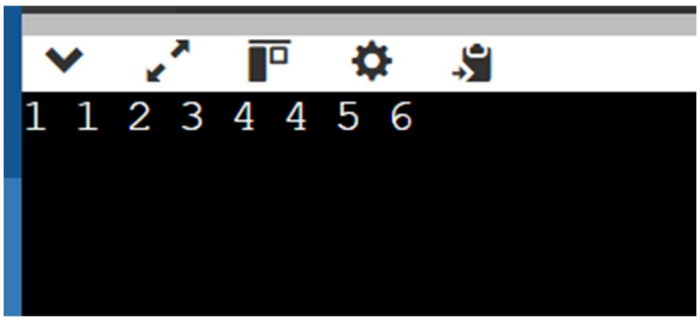
CODE:

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

struct ListNode {
    int val;
    ListNode* next;
    ListNode() : val(0), next(nullptr) {}
    ListNode(int x) : val(x), next(nullptr) {}
    ListNode(int x, ListNode* next) : val(x), next(next) {}
};

struct Compare {
    bool operator()(ListNode* a, ListNode* b) {
        return a->val > b->val;
```

```
    }
};

ListNode* mergeKLists(vector<ListNode*>& lists) {
    priority_queue<ListNode*, vector<ListNode*>, Compare> minHeap;

    for (ListNode* list : lists) {
        if (list) {
            minHeap.push(list);
        }
    }

    ListNode* dummy = new ListNode(-1);
    ListNode* current = dummy;

    while (!minHeap.empty()) {
        ListNode* smallest = minHeap.top();
        minHeap.pop();

        current->next = smallest;
        current = current->next;

        if (smallest->next) {
            minHeap.push(smallest->next);
        }
    }

    return dummy->next;
}

ListNode* createList(const vector<int>& values) {
    ListNode* dummy = new ListNode(-1);
    ListNode* current = dummy;
```

```
    for (int val : values) {
        current->next = new ListNode(val);
        current = current->next;
    }
    return dummy->next;
}

void printList(ListNode* head) {
    while (head) {
        cout << head->val << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    vector<ListNode*> lists1 = {
        createList({1, 4, 5}),
        createList({1, 3, 4}),
        createList({2, 6})
    };
    ListNode* result1 = mergeKLists(lists1);
    printList(result1);

    vector<ListNode*> lists2 = {};
    ListNode* result2 = mergeKLists(lists2);
    printList(result2);

    vector<ListNode*> lists3 = {createList({})};
    ListNode* result3 = mergeKLists(lists3);
    printList(result3);
    return 0;
}
```

OUTPUT:



Problem-5(VeryHard)

CODE:

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

double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    if (nums1.size() > nums2.size()) {
        return findMedianSortedArrays(nums2, nums1);
    }

    int m = nums1.size();
    int n = nums2.size();
    int left = 0, right = m;
    int halfLen = (m + n + 1) / 2;

    while (left <= right) {
        int i = (left + right) / 2;
        int j = halfLen - i;

        int nums1LeftMax = (i == 0) ? INT_MIN : nums1[i - 1];
```

```
        int nums1RightMin = (i == m) ? INT_MAX : nums1[i];
        int nums2LeftMax = (j == 0) ? INT_MIN : nums2[j - 1];
        int nums2RightMin = (j == n) ? INT_MAX : nums2[j];

        if (nums1LeftMax <= nums2RightMin && nums2LeftMax <=
nums1RightMin) {
            if ((m + n) % 2 == 0) {
                return (max(nums1LeftMax, nums2LeftMax) + min(nums1RightMin,
nums2RightMin)) / 2.0;
            } else {
                return max(nums1LeftMax, nums2LeftMax);
            }
        } else if (nums1LeftMax > nums2RightMin) {
            right = i - 1;
        } else {
            left = i + 1;
        }
    }

    throw runtime_error("Input arrays are not sorted.");
}

int main() {
    vector<int> nums1 = {1, 3};
    vector<int> nums2 = {2};
    cout << "Median: " << findMedianSortedArrays(nums1, nums2) << endl;

    vector<int> nums1_2 = {1, 2};
    vector<int> nums2_2 = {3, 4};
    cout << "Median: " << findMedianSortedArrays(nums1_2, nums2_2) << endl;

    return 0;
}
```

OUTPUT:

```
Median: 2
Median: 2.5


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