main.cpp

```cpp
#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 k = 16;
    vector<int> arr = {9, 7, 16, 16, 4};
    int result = findFirstOccurrence(k, arr);
    cout << result << endl;
    return 0;
}
```

```
3


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

main.cpp

```cpp
#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 totalMoves = 0;

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

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

input

```
3


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

Run | Debug | Stop | Share | Save | { } Beautify

main.cpp

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

bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int m = matrix.size();
    int n = matrix[0].size();

    int left = 0, right = m * n - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        int mid_value = matrix[mid / n][mid % n];

        if (mid_value == target) {
            return true;
        } else if (mid_value < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return false;
}

int main() {
    vector<vector<int>> matrix = {
        {1, 3, 5, 7},
        {10, 11, 16, 20},
        {23, 30, 34, 60}
```

input

```
True


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

main.cpp

```cpp
#include <iostream>
#include <vector>
#include <queue>
#include <unordered_map>
#include <unordered_set>

using namespace std;

vector<int> topologicalSort(int n, vector<vector<int>>& graph, vector<int>& inDegree) {
    queue<int> q;
    vector<int> sortedList;
    for (int i = 0; i < n; ++i) {
        if (inDegree[i] == 0) {
            q.push(i);
        }
    }
    while (!q.empty()) {
        int node = q.front();
        q.pop();
        sortedList.push_back(node);

        for (int neighbor : graph[node]) {
            --inDegree[neighbor];
            if (inDegree[neighbor] == 0) {
                q.push(neighbor);
            }
        }
    }
    if (sortedList.size() != n) {
        return {};
```

input

```
No valid sorting found.


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

main.cpp

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

int findMin(vector<int>& nums) {
    int left = 0, right = nums.size() - 1;

    while (left < right) {
        int mid = left + (right - left) / 2;

        // If the middle element is equal to the rightmost element, reduce the search space
        if (nums[mid] == nums[right]) {
            right--;
        }
        // If mid element is greater than rightmost, the minimum is in the right part
        else if (nums[mid] > nums[right]) {
            left = mid + 1;
        }
        // If mid element is smaller than rightmost, the minimum is in the left part
        else {
            right = mid;
        }
    }

    return nums[left];  // Left will point to the minimum element
}

int main() {
    vector<int> nums = {3, 2, 7, 8, 1};
    cout << "Minimum element: " << findMin(nums) << endl; // Output: 0
```

input

```
Minimum element: 1


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