```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <limits>

using namespace std;

// FCFS Disk Scheduling
void FCFS(const vector<int>& requests, int head) {
    int seekTime = 0;
    cout << "Execution order: " << head;
    for (int i = 0; i < requests.size(); i++) {
        seekTime += abs(requests[i] - head);
        head = requests[i];
        cout << " -> " << head;
    }
    cout << "\nTotal seek time: " << seekTime << endl;
}

// SSTF Disk Scheduling
void SSTF(vector<int> requests, int head) {
    int seekTime = 0;
    vector<bool> visited(requests.size(), false);
    cout << "Execution order: " << head;

    for (size_t i = 0; i < requests.size(); i++) {
        int minDistance = numeric_limits<int>::max();
        int index = -1;

        for (size_t j = 0; j < requests.size(); j++) {
            if (!visited[j] && abs(requests[j] - head) < minDistance) {
                minDistance = abs(requests[j] - head);
                index = j;
            }
        }

        if (index != -1) {
            visited[index] = true;
            seekTime += minDistance;
            head = requests[index];
            cout << " -> " << head;
        }
    }
    cout << "\nTotal seek time: " << seekTime << endl;
}

// SCAN Disk Scheduling
void SCAN(vector<int> requests, int head, int diskSize) {
    int seekTime = 0;
    vector<int> left, right;

    // Separate requests into left and right of the head
    for (int req : requests) {
        if (req < head) {
            left.push_back(req);
        } else {
            right.push_back(req);
        }
    }

    // Add boundaries (0 and diskSize - 1)
    left.push_back(0);
    right.push_back(diskSize - 1);

    // Sort the requests
    sort(left.begin(), left.end(), greater<int>()); // Descending for left
    sort(right.begin(), right.end());               // Ascending for right

    cout << "Execution order: " << head;

    // Move towards the right
    for (int req : right) {
        seekTime += abs(req - head);
        head = req;
        cout << " -> " << head;
    }

    // Move towards the left
    for (int req : left) {
        seekTime += abs(req - head);
        head = req;
        cout << " -> " << head;
    }

    cout << "\nTotal seek time: " << seekTime << endl;
}

int main() {
    int n;
    cout << "Enter number of requests: ";
    cin >> n;

    if (cin.fail() || n <= 0) {
        cout << "Invalid input! Number of requests must be a positive integer." << endl;
        return 1;
    }

    vector<int> requests(n);
    cout << "Enter request queue: ";
    for (int i = 0; i < n; i++) {
        cin >> requests[i];
        if (cin.fail()) {
            cout << "Invalid input! Please enter integers only." << endl;
            return 1;
        }
    }

    int head;
    cout << "Enter initial head position: ";
    cin >> head;

    if (cin.fail()) {
        cout << "Invalid input! Please enter an integer." << endl;
        return 1;
    }

    int diskSize;
    cout << "Enter disk size: ";
    cin >> diskSize;

    if (cin.fail() || diskSize <= 0) {
        cout << "Invalid input! Disk size must be a positive integer." << endl;
        return 1;
    }

    int choice;
    cout << "Choose Algorithm:\n1. FCFS\n2. SSTF\n3. SCAN\nEnter choice: ";
    cin >> choice;

    if (cin.fail()) {
        cout << "Invalid input! Please enter an integer." << endl;
        return 1;
    }

    switch (choice) {
        case 1:
            FCFS(requests, head);
            break;
        case 2:
            SSTF(requests, head);
            break;
        case 3:
            SCAN(requests, head, diskSize);
            break;
        default:
            cout << "Invalid choice!" << endl;
            break;
    }

    return 0;
}
```