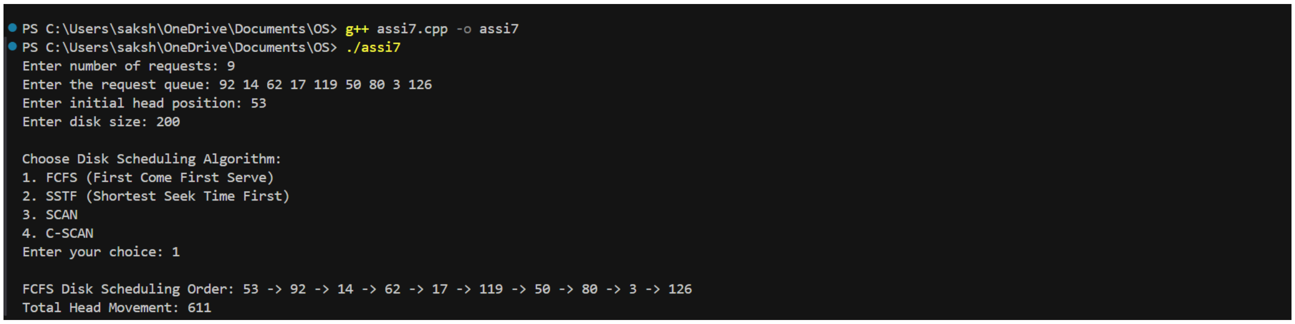
Name-Sujal Santosh Porte
PRN-122B1B227

Assignment No-8

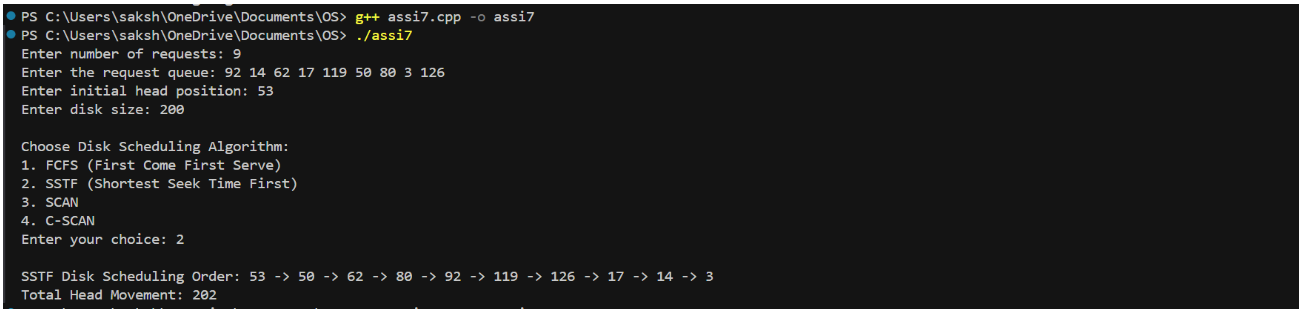
Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <climits>

using namespace std;
void FCFS(vector<int> requests, int head);
void SSTF(vector<int> requests, int head);
void SCAN(vector<int> requests, int head, int disk_size, string direction);
void C_SCAN(vector<int> requests, int head, int disk_size, string direction);

int main() {
    int n, head, disk_size, choice;
    string direction;
    cout << "Enter number of requests: ";
    cin >> n;

    vector<int> requests(n);
    cout << "Enter the request queue: ";
    for (int i = 0; i < n; i++) {
        cin >> requests[i];
    }

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

    cout << "Enter disk size: ";
    cin >> disk_size;

    cout << "\nChoose Disk Scheduling Algorithm:\n";
    cout << "1. FCFS (First Come First Serve)\n";
    cout << "2. SSTF (Shortest Seek Time First)\n";
    cout << "3. SCAN\n";
    cout << "4. C-SCAN\n";
    cout << "Enter your choice: ";
    cin >> choice;
    if (choice == 3 || choice == 4) {
        cout << "Enter direction (left/right): ";
        cin >> direction;
    }

    switch (choice) {
        case 1:
            FCFS(requests, head);
            break;
        case 2:
            SSTF(requests, head);
            break;
        case 3:
            SCAN(requests, head, disk_size, direction);
            break;
        case 4:
            C_SCAN(requests, head, disk_size, direction);
            break;
        default:
            cout << "Invalid choice!" << endl;
    }
    return 0;
}


void FCFS(vector<int> requests, int head) {
    int total_head_movement = 0;
    cout << "\nFCFS Disk Scheduling Order: " << head;

    for (int request : requests) {
        total_head_movement += abs(request - head);
        head = request;
        cout << " -> " << request;
    }

    cout << "\nTotal Head Movement: " << total_head_movement << endl;
}
void SSTF(vector<int> requests, int head) {
    int total_head_movement = 0;
    vector<bool> visited(requests.size(), false);

    cout << "\nSSTF Disk Scheduling Order: " << head;

    for (size_t i = 0; i < requests.size(); i++) {
        int min_distance = INT_MAX, index = -1;
        for (size_t j = 0; j < requests.size(); j++) {
            if (!visited[j] && abs(requests[j] - head) < min_distance) {
                min_distance = abs(requests[j] - head);
                index = j;
            }
        }
        if (index == -1) break; // If no unvisited request is left
        visited[index] = true;
        total_head_movement += min_distance;
        head = requests[index];
        cout << " -> " << requests[index];
    }

    cout << "\nTotal Head Movement: " << total_head_movement << endl;
}

// 3. SCAN (Elevator Algorithm)
void SCAN(vector<int> requests, int head, int disk_size, string direction) {
    int total_head_movement = 0;
    vector<int> left, right;

    for (int req : requests) {
        if (req < head)
            left.push_back(req);
        else
            right.push_back(req);
    }

    sort(left.begin(), left.end());
    sort(right.begin(), right.end());

    cout << "\nSCAN Disk Scheduling Order: " << head;

    if (direction == "left") {
        for (int i = left.size() - 1; i >= 0; i--) {
            total_head_movement += abs(left[i] - head);
            head = left[i];
            cout << " -> " << left[i];
        }
        total_head_movement += head;
        cout << " -> 0";
        head = 0;
        for (int req : right) {
            total_head_movement += abs(req - head);
            head = req;
            cout << " -> " << req;
        }
    } else { // Right direction
        for (int req : right) {
            total_head_movement += abs(req - head);
            head = req;
            cout << " -> " << req;
        }
        total_head_movement += abs((disk_size - 1) - head);
        cout << " -> " << disk_size - 1;
        head = disk_size - 1;

        for (int i = left.size() - 1; i >= 0; i--) {
            total_head_movement += abs(left[i] - head);
            head = left[i];
            cout << " -> " << left[i];
        }
    }

    cout << "\nTotal Head Movement: " << total_head_movement << endl;
}

// 4. C-SCAN (Circular SCAN)
void C_SCAN(vector<int> requests, int head, int disk_size, string direction) {
    int total_head_movement = 0;
    vector<int> left, right;

    for (int req : requests) {
        if (req < head)
            left.push_back(req);
        else
            right.push_back(req);
    }

    sort(left.begin(), left.end());
    sort(right.begin(), right.end());

    cout << "\nC-SCAN Disk Scheduling Order: " << head;

    if (direction == "left") {
        for (int i = left.size() - 1; i >= 0; i--) {
            total_head_movement += abs(left[i] - head);
            head = left[i];
            cout << " -> " << left[i];
        }
        total_head_movement += head;
        cout << " -> 0";
        head = 0;

        total_head_movement += (disk_size - 1);
        cout << " -> " << disk_size - 1;
        head = disk_size - 1;

        for (int i = right.size() - 1; i >= 0; i--) {
            total_head_movement += abs(right[i] - head);
            head = right[i];
            cout << " -> " << right[i];
        }
    } else { // Right direction
        for (int req : right) {
            total_head_movement += abs(req - head);
            head = req;
            cout << " -> " << req;
        }
        total_head_movement += abs((disk_size - 1) - head);
        cout << " -> " << disk_size - 1;
        head = disk_size - 1;

        total_head_movement += abs(0 - head);
        cout << " -> 0";
        head = 0;

        for (int req : left) {
            total_head_movement += abs(req - head);
            head = req;
            cout << " -> " << req;
        }
    }

    cout << "\nTotal Head Movement: " << total_head_movement << endl;
}
```

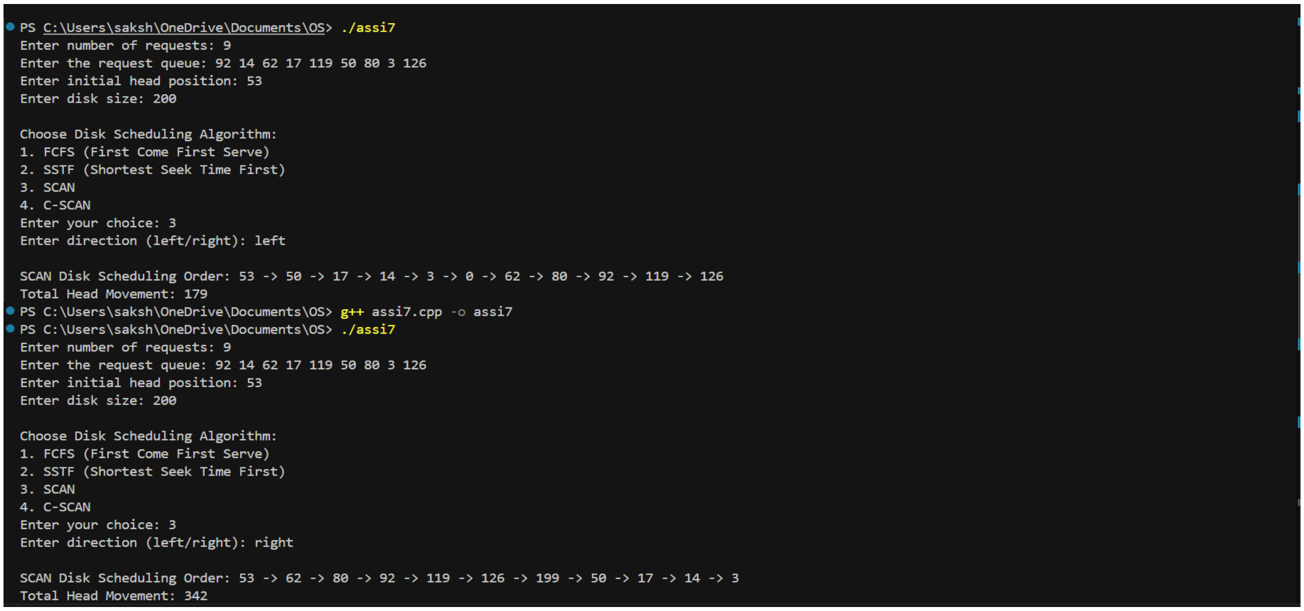
Output:



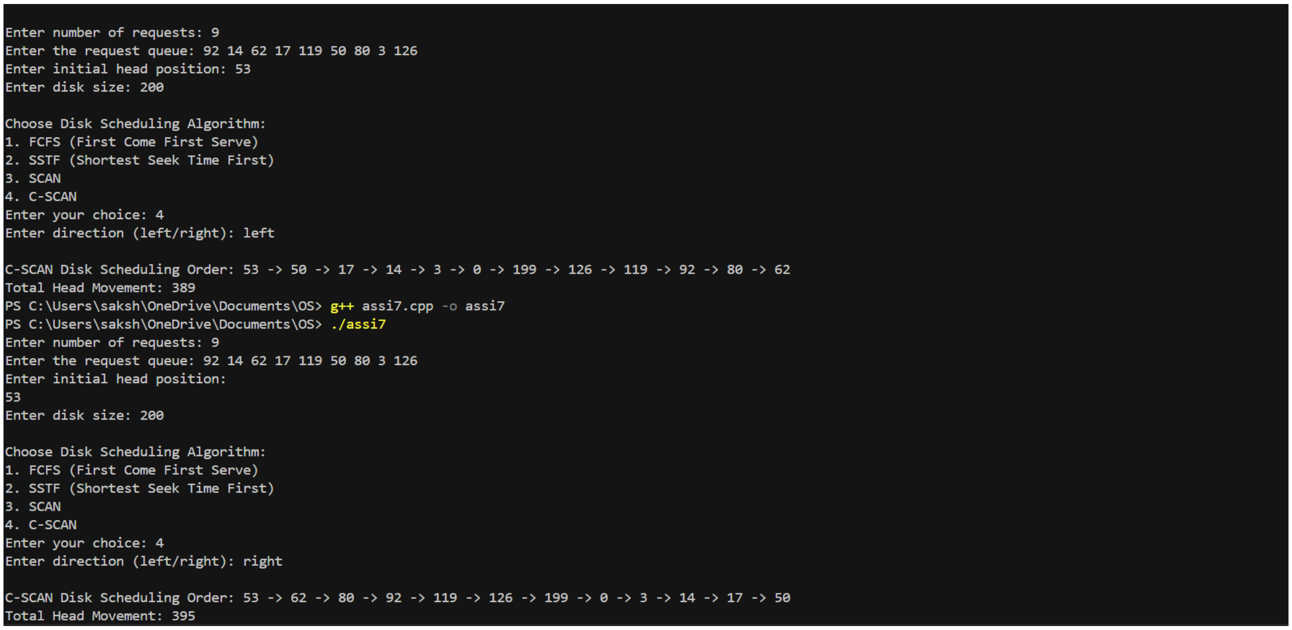
```
PS C:\Users\saksh\OneDrive\Documents\OS> g++ assi7.cpp -o assi7
PS C:\Users\saksh\OneDrive\Documents\OS> ./assi7
Enter number of requests: 9
Enter the request queue: 92 14 62 17 119 50 80 3 126
Enter initial head position: 53
Enter disk size: 200

Choose Disk Scheduling Algorithm:
1. FCFS (First Come First Serve)
2. SSTF (Shortest Seek Time First)
3. SCAN
4. C-SCAN
Enter your choice: 1

FCFS Disk Scheduling Order: 53 -> 92 -> 14 -> 62 -> 17 -> 119 -> 50 -> 80 -> 3 -> 126
Total Head Movement: 611
```



```
PS C:\Users\saksh\OneDrive\Documents\OS> g++ assi7.cpp -o assi7
PS C:\Users\saksh\OneDrive\Documents\OS> ./assi7
Enter number of requests: 9
Enter the request queue: 92 14 62 17 119 50 80 3 126
Enter initial head position: 53
Enter disk size: 200

Choose Disk Scheduling Algorithm:
1. FCFS (First Come First Serve)
2. SSTF (Shortest Seek Time First)
3. SCAN
4. C-SCAN
Enter your choice: 2

SSTF Disk Scheduling Order: 53 -> 50 -> 62 -> 80 -> 92 -> 119 -> 126 -> 17 -> 14 -> 3
Total Head Movement: 202
```



```
PS C:\Users\saksh\OneDrive\Documents\OS> ./assi7
Enter number of requests: 9
Enter the request queue: 92 14 62 17 119 50 80 3 126
Enter initial head position: 53
Enter disk size: 200

Choose Disk Scheduling Algorithm:
1. FCFS (First Come First Serve)
2. SSTF (Shortest Seek Time First)
3. SCAN
4. C-SCAN
Enter your choice: 3
Enter direction (left/right): left

SCAN Disk Scheduling Order: 53 -> 50 -> 17 -> 14 -> 3 -> 0 -> 62 -> 80 -> 92 -> 119 -> 126
Total Head Movement: 179
PS C:\Users\saksh\OneDrive\Documents\OS> g++ assi7.cpp -o assi7
PS C:\Users\saksh\OneDrive\Documents\OS> ./assi7
Enter number of requests: 9
Enter the request queue: 92 14 62 17 119 50 80 3 126
Enter initial head position: 53
Enter disk size: 200

Choose Disk Scheduling Algorithm:
1. FCFS (First Come First Serve)
2. SSTF (Shortest Seek Time First)
3. SCAN
4. C-SCAN
Enter your choice: 3
Enter direction (left/right): right

SCAN Disk Scheduling Order: 53 -> 62 -> 80 -> 92 -> 119 -> 126 -> 199 -> 50 -> 17 -> 14 -> 3
Total Head Movement: 342
```



```
Enter number of requests: 9
Enter the request queue: 92 14 62 17 119 50 80 3 126
Enter initial head position: 53
Enter disk size: 200

Choose Disk Scheduling Algorithm:
1. FCFS (First Come First Serve)
2. SSTF (Shortest Seek Time First)
3. SCAN
4. C-SCAN
Enter your choice: 4
Enter direction (left/right): left

C-SCAN Disk Scheduling Order: 53 -> 50 -> 17 -> 14 -> 3 -> 0 -> 199 -> 126 -> 119 -> 92 -> 80 -> 62
Total Head Movement: 389
PS C:\Users\saksh\OneDrive\Documents\OS> g++ assi7.cpp -o assi7
PS C:\Users\saksh\OneDrive\Documents\OS> ./assi7
Enter number of requests: 9
Enter the request queue: 92 14 62 17 119 50 80 3 126
Enter initial head position:
53
Enter disk size: 200

Choose Disk Scheduling Algorithm:
1. FCFS (First Come First Serve)
2. SSTF (Shortest Seek Time First)
3. SCAN
4. C-SCAN
Enter your choice: 4
Enter direction (left/right): right

C-SCAN Disk Scheduling Order: 53 -> 62 -> 80 -> 92 -> 119 -> 126 -> 199 -> 0 -> 3 -> 14 -> 17 -> 50
Total Head Movement: 395
```