Experiment Number: 11

Problem Statement: **Implementation of following Disk Scheduling algorithms : FCFS, SCAN, C-SCAN, SSTF.LOOK  and C-LOOK**

NAME: Aadesh Chawla ROLLNO: 12

CLASS: TY-IT-A BATCH: B1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code:**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void fcfs(const   vector<int>& requests, int head) {

    int seek\_time = 0;

    int curr\_track;

    for (const int& request : requests) {

        curr\_track = request;

        seek\_time +=   abs(curr\_track - head);

        head = curr\_track;

    }

      cout << "FCFS Total Seek Time: " << seek\_time <<   endl;

      cout << "FCFS Average Seek Time: " << static\_cast<float>(seek\_time) / requests.size() <<   endl;

}

void scan(const   vector<int>& requests, int head, int max\_track) {

    int seek\_time = 0;

    int curr\_track;

    int direction = 1; // 1 for right, -1 for left

    int index = 0;

      vector<int> temp(requests);

    temp.push\_back(head);

    int size = temp.size();

      sort(temp.begin(), temp.end());

    for (int i = 0; i < size; i++) {

        if (temp[i] == head) {

            index = i;

            break;

        }

    }

    if (direction == 1) {

        for (int i = index; i < size; i++) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

        seek\_time +=   abs(max\_track - head);

        head = max\_track;

        for (int i = index - 1; i >= 0; i--) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

    } else {

        for (int i = index; i >= 0; i--) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

        seek\_time +=   abs(0 - head);

        head = 0;

        for (int i = index + 1; i < size; i++) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

    }

      cout << "SCAN Total Seek Time: " << seek\_time <<   endl;

      cout << "SCAN Average Seek Time: " << static\_cast<float>(seek\_time) / requests.size() <<   endl;

}

void c\_scan(const   vector<int>& requests, int head, int max\_track) {

    int seek\_time = 0;

    int curr\_track;

    int direction = 1; // 1 for right, -1 for left

    int index = 0;

      vector<int> temp(requests);

    temp.push\_back(0);

    temp.push\_back(max\_track);

      sort(temp.begin(), temp.end());

    int size = temp.size();

    for (int i = 0; i < size; i++) {

        if (temp[i] == head) {

            index = i;

            break;

        }

    }

    if (direction == 1) {

        for (int i = index; i < size; i++) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

        seek\_time +=   abs(max\_track - head);

        head = max\_track;

        for (int i = 0; i < index - 1; i++) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

    } else {

        for (int i = index; i >= 0; i--) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

        seek\_time +=   abs(0 - head);

        head = 0;

        for (int i = size - 2; i > index; i--) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

    }

      cout << "C-SCAN Total Seek Time: " << seek\_time <<   endl;

      cout << "C-SCAN Average Seek Time: " << static\_cast<float>(seek\_time) / requests.size() <<   endl;

}

void sstf(const   vector<int>& requests, int head) {

    int seek\_time = 0;

    int curr\_track;

      vector<int> temp(requests);

    for (size\_t i = 0; i < temp.size(); i++) {

        size\_t min\_index = i;

        for (size\_t j = i + 1; j < temp.size(); j++) {

            if (  abs(temp[j] - head) <   abs(temp[min\_index] - head)) {

                min\_index = j;

            }

        }

        curr\_track = temp[min\_index];

        seek\_time +=   abs(curr\_track - head);

        head = curr\_track;

        temp[min\_index] = temp[i];

        temp[i] = curr\_track;

    }

      cout << "SSTF Total Seek Time: " << seek\_time <<   endl;

      cout << "SSTF Average Seek Time: " << static\_cast<float>(seek\_time) / requests.size() <<   endl;

}

void look(const   vector<int>& requests, int head, int max\_track) {

    int seek\_time = 0;

    int curr\_track;

    int direction = 1; // 1 for right, -1 for left

    int index = 0;

      vector<int> temp(requests);

      sort(temp.begin(), temp.end());

    int size = temp.size();

    for (int i = 0; i < size; i++) {

        if (temp[i] == head) {

            index = i;

            break;

        }

    }

    if (direction == 1) {

        for (int i = index; i < size; i++) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

        for (int i = size - 2; i >= index; i--) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

    } else {

        for (int i = index; i >= 0; i--) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

        for (int i = 1; i <= index; i++) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

    }

      cout << "LOOK Total Seek Time: " << seek\_time <<   endl;

      cout << "LOOK Average Seek Time: " << static\_cast<float>(seek\_time) / requests.size() <<   endl;

}

void c\_look(const   vector<int>& requests, int head) {

    int seek\_time = 0;

    int curr\_track;

    int direction = 1; // 1 for right, -1 for left

    int index = 0;

      vector<int> temp(requests);

    temp.push\_back(head);

      sort(temp.begin(), temp.end());

    int size = temp.size();

    for (int i = 0; i < size; i++) {

        if (temp[i] == head) {

            index = i;

            break;

        }

    }

    if (direction == 1) {

        for (int i = index; i < size; i++) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

        for (int i = 0; i < index - 1; i++) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

    } else {

        for (int i = index; i >= 0; i--) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

        for (int i = size - 2; i >= index; i--) {

            curr\_track = temp[i];

            seek\_time +=   abs(curr\_track - head);

            head = curr\_track;

        }

    }

      cout << "C-LOOK Total Seek Time: " << seek\_time <<   endl;

      cout << "C-LOOK Average Seek Time: " << static\_cast<float>(seek\_time) / requests.size() <<   endl;

}

int main() {

    int size, head, max\_track;

      cout << "Enter the number of requests: ";

      cin >> size;

      vector<int> arr(size);

      cout << "Enter the requests: ";

    for (int i = 0; i < size; i++) {

          cin >> arr[i];

    }

      cout << "Enter the initial head position: ";

      cin >> head;

      cout << "Enter the maximum track number: ";

      cin >> max\_track;

      cout <<   endl;

      cout << "FCFS:" <<   endl;

    fcfs(arr, head);

      cout <<   endl;

      cout << "SCAN:" <<   endl;

    scan(arr, head, max\_track);

      cout <<   endl;

      cout << "C-SCAN:" <<   endl;

    c\_scan(arr, head, max\_track);

      cout <<   endl;

      cout << "SSTF:" <<   endl;

    sstf(arr, head);

      cout <<   endl;

      cout << "LOOK:" <<   endl;

    look(arr, head, max\_track);

      cout <<   endl;

      cout << "C-LOOK:" <<   endl;

    c\_look(arr, head);

    return 0;

}

Output:

