```cpp
#include <bits/stdc++.h>
using namespace std;
void fifo(vector<int> requests, int head) {
 cout << head << " -> ";
 int cost = 0;
 int newHead = head;
 int count = 0;
 for (int i = 0; i < requests.size() - 1; i++)
 {
 cout << requests[i] << " -> ";
 cost += abs(newHead - requests[i]);
 if (newHead != requests[i])
 count++;
 newHead = requests[i];
 }
 cout << requests.back() << endl;
 count++;
 cost += abs(newHead - requests.back());
 cout << "Total head movements = " << cost << endl;
 cout << "Avegrage head movements = " << (double)cost / count << endl; }
void ascending(vector<int> requests, int head, int *count, int *cost, int *newHead) {
 vector<int>::iterator headLoc = find(requests.begin(), requests.end(), head);
 for (auto i = headLoc - 1; i >= requests.begin(); i--)
 {
 cout << *i << " -> ";
 *cost += abs(*newHead - *i);
 if (*newHead != *i)
 *count += 1;
 *newHead = *i;
 }
}
void descending(vector<int> requests, int head, int *count, int *cost, int *newHead) {
 vector<int>::iterator headLoc = find(requests.begin(), requests.end(), head);
 for (auto i = headLoc + 1; i < requests.end(); i++)
 {
 cout << *i << " -> ";
 *cost += abs(*newHead - *i);
 if (*newHead != *i)
 *count += 1;
 *newHead = *i;
 }
}
void cscan(vector<int> requests, int head, bool isAscending) {
 requests.push_back(head);
 sort(requests.begin(), requests.end());
 vector<int>::iterator headLoc = find(requests.begin(), requests.end(), head);
 int cost = 0;
 int newHead = head;
 int count = 0;
 if (isAscending)
 {
 for (auto i = headLoc; i >= requests.begin(); i--)
 {
 cout << *i << " -> ";
 cost += abs(newHead - *i);
 if (newHead != *i)
 count += 1;
 newHead = *i;
 }
 for (auto i = requests.end() - 1; i > headLoc; i--)
 {
 cout << *i << " -> ";
 cost += abs(newHead - *i);
 if (newHead != *i)
 count += 1;
 newHead = *i;
 }
 }
 else
 {
 for (auto i = headLoc; i < requests.end(); i++)
 {
 cout << *i << " -> ";
 cost += abs(newHead - *i);
 if (newHead != *i)
 count += 1;
 newHead = *i;
 }
 for (auto i = requests.begin(); i < headLoc; i++)
 {
 cout << *i << " -> ";
 cost += abs(newHead - *i);
 if (newHead != *i)
 count += 1;
 newHead = *i;
 }
 }
 cout << endl
 << "Total head movements = " << cost << endl;
 cout << "Avegrage head movements = " << (double)cost / count << endl; }
void scan(vector<int> requests, int head, bool isAscending) {
 requests.push_back(head);
 sort(requests.begin(), requests.end());
 vector<int>::iterator headLoc = find(requests.begin(), requests.end(), head);
 int cost = 0;
 int newHead = head;
 int count = 0;
 cout << head << " -> ";
 if (isAscending)
 {
 ascending(requests, head, &count, &cost, &newHead);
 descending(requests, head, &count, &cost, &newHead);
 }
 else
 {
 descending(requests, head, &count, &cost, &newHead);
 ascending(requests, head, &count, &cost, &newHead);
 }
 cout << endl
 << "Total head movements = " << cost << endl;
 cout << "Avegrage head movements = " << (double)cost / count << endl; }
int main()
{
 int n, size;
 cout << "please enter size of disk: ";
 cin >> size;
 cout << "please enter number of requests: ";
 cin >> n;
 cout << "please enter the requests: ";
 vector<int> requests;
 for (int i = 0; i < n; i++)
 {
 int temp;
 cin >> temp;
 requests.push_back(temp);
 }
 int head;
 cout << "please enter head location: ";
 cin >> head;
 cout << "please choose the direction (1- ascending, 2- descending): ";
 int isAscending;
 cin >> isAscending;
 cout << "FIFO" << endl;
 fifo(requests, head);
 cout << "SCAN" << endl;
 scan(requests, head, isAscending == 2 ? 1 : 0);
 cout << "CSCAN" << endl;
 cscan(requests, head, isAscending == 2 ? 1 : 0);
 return 0;
}
```