# Name: Rasika Kailas Mate PRN No.: 122B2B293

**Title: Write a program to implement disk scheduling algorithms FIFO, SSTF, SCAN, C- SCAN**

# ----------------------------------------------------------------------------------------------------------------

**Code:**

#include <stdio.h> #include <stdlib.h> #include <limits.h>

#define MAX\_REQUESTS 100

void fifo(int requests[], int head, int num\_requests) { int total\_seek\_time = 0;

int current\_track = head;

for (int i = 0; i < num\_requests; i++) { total\_seek\_time += abs(current\_track - requests[i]); current\_track = requests[i];

}

printf("\nFIFO\nTotal Seek Time: %d\n", total\_seek\_time);

}

void sstf(int requests[], int head, int num\_requests) { int total\_seek\_time = 0;

int current\_track = head;

int \*visited = (int \*)calloc(num\_requests, sizeof(int)); int min\_distance, next\_track;

for (int i = 0; i < num\_requests; i++) { min\_distance = INT\_MAX;

for (int j = 0; j < num\_requests; j++) {

if (!visited[j] && abs(current\_track - requests[j]) < min\_distance) { next\_track = j;

min\_distance = abs(current\_track - requests[j]);

}

}

total\_seek\_time += min\_distance; current\_track = requests[next\_track]; visited[next\_track] = 1;

}

printf("\nSSTF\nTotal Seek Time: %d\n", total\_seek\_time); free(visited);

}

void scan(int requests[], int head, int num\_requests, int direction) { int total\_seek\_time = 0;

int current\_track = head;

int \*visited = (int \*)calloc(num\_requests, sizeof(int)); int start\_track = 0;

int end\_track = 199; if (direction == -1) { start\_track = 199;

end\_track = 0;

}

while (1) {

int next\_track = -1;

for (int i = 0; i < num\_requests; i++) {

if (!visited[i] && requests[i] >= start\_track && requests[i] <= end\_track) { if (next\_track == -1 || abs(current\_track - requests[i]) < abs(current\_track -

requests[next\_track])) {

next\_track = i;

}

}

}

if (next\_track == -1) { if (direction == 1) { end\_track = 199;

} else {

end\_track = 0;

}

direction \*= -1;

} else {

total\_seek\_time += abs(current\_track - requests[next\_track]); current\_track = requests[next\_track];

visited[next\_track] = 1;

}

if (next\_track == -1 && visited[num\_requests - 1]) { break;

}

}

printf("\nSCAN\nTotal Seek Time: %d\n", total\_seek\_time); free(visited);

}

void c\_scan(int requests[], int head, int num\_requests, int direction) { int total\_seek\_time = 0;

int current\_track = head;

int \*visited = (int \*)calloc(num\_requests, sizeof(int)); int start\_track = 0;

int end\_track = 199; if (direction == -1) { start\_track = 199;

end\_track = 0;

}

while (1) {

int next\_track = -1;

for (int i = 0; i < num\_requests; i++) {

if (!visited[i] && requests[i] >= start\_track && requests[i] <= end\_track) { if (next\_track == -1 || abs(current\_track - requests[i]) < abs(current\_track -

requests[next\_track])) {

next\_track = i;

}

}

}

if (next\_track == -1) { if (direction == 1) {

total\_seek\_time += abs(current\_track - 199); current\_track = 199;

start\_track = 199;

end\_track = 0;

} else {

total\_seek\_time += abs(current\_track - 0); current\_track = 0;

start\_track = 0;

end\_track = 199;

}

} else {

total\_seek\_time += abs(current\_track - requests[next\_track]); current\_track = requests[next\_track];

visited[next\_track] = 1;

}

if (next\_track == -1 && visited[num\_requests - 1]) { break;

}

}

printf("\nC-SCAN\nTotal Seek Time: %d\n", total\_seek\_time); free(visited);

}

int main() {

int num\_requests, head;

int requests[MAX\_REQUESTS]; printf("Enter total number of requests: "); scanf("%d", &num\_requests); printf("Enter the requests: ");

for (int i = 0; i < num\_requests; i++) { scanf("%d", &requests[i]);

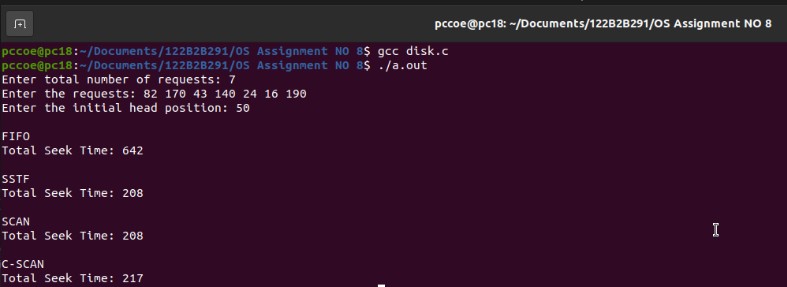
}

printf("Enter the initial head position: "); scanf("%d", &head);

fifo(requests, head, num\_requests); sstf(requests, head, num\_requests);

scan(requests, head, num\_requests, 1); // Direction: 1 (Towards higher tracks) c\_scan(requests, head, num\_requests, 1); // Direction: 1 (Towards higher tracks) return 0;

}

**Output:**