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: