```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX_REQUESTS 100

// Function to calculate absolute difference
int abs_diff(int a, int b) {
    return abs(a - b);
}

// Function to sort the request array
void sort(int arr[], int n) {
    int temp;
    for (int i = 0; i < n -1; i++) {
        for (int j = 0; j < n - i -1; j++) {
            if (arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

// SSTF Algorithm
void SSTF(int requests[], int n, int head) {
    int seek_count = 0;
```

```
int distance, cur_track;
int visited[MAX_REQUESTS] = {0};
int count = 0;

printf("\nSSTF Disk Scheduling Algorithm\n");
printf("Seek Sequence: ");

while (count < n) {
   int min = 1e9;
   int index = -1;
   for (int i = 0; i < n; i++) {
      if (!visited[i]) {
         distance = abs_diff(head, requests[i]);
         if (distance < min) {
            min = distance;
            index = i;
         }
      }
   }
   visited[index] = 1;
   seek_count += abs_diff(head, requests[index]);
   head = requests[index];
   printf("%d ", head);
   count++;
}

printf("\nTotal Seek Operations: %d\n", seek_count);
printf("Average Seek Time: %.2f\n", (float)seek_count / n);
```

```
}

// SCAN Algorithm
void SCAN(int requests[], int n, int head, int disk_size) {
    int seek_count = 0;
    int distance, cur_track;
    int direction = 1; // 1 for moving towards higher cylinder numbers
    int size = n + 1;
    int temp_requests[MAX_REQUESTS];

    for (int i = 0; i < n; i++)
        temp_requests[i] = requests[i];
    temp_requests[n] = head;
    sort(temp_requests, size);

    int index;
    for (int i = 0; i < size; i++) {
        if (temp_requests[i] == head) {
            index = i;
            break;
        }
    }

    printf("\nSCAN Disk Scheduling Algorithm\n");
    printf("Seek Sequence: ");

    // Move towards higher cylinder numbers
    for (int i = index + 1; i < size; i++) {
```

```
        seek_count += abs_diff(head, temp_requests[i]);
        head = temp_requests[i];
        printf("%d ", head);
    }

    // If the head is not at the end, move to the end
    if (head != disk_size - 1) {
        seek_count += abs_diff(head, disk_size - 1);
        head = disk_size - 1;
        printf("%d ", head);
    }

    // Reverse direction and move towards lower cylinder numbers
    for (int i = index - 1; i >= 0; i--) {
        seek_count += abs_diff(head, temp_requests[i]);
        head = temp_requests[i];
        printf("%d ", head);
    }

    printf("\nTotal Seek Operations: %d\n", seek_count);
    printf("Average Seek Time: %.2f\n", (float)seek_count / n);
}

// C-LOOK Algorithm
void C_LOOK(int requests[], int n, int head) {
    int seek_count = 0;
    int distance, cur_track;
    int size = n + 1;
```

```
int temp_requests[MAX_REQUESTS];

for (int i = 0; i < n; i++)
    temp_requests[i] = requests[i];
temp_requests[n] = head;
sort(temp_requests, size);

int index;
for (int i = 0; i < size; i++) {
    if (temp_requests[i] == head) {
        index = i;
        break;
    }
}

printf("\nC-LOOK Disk Scheduling Algorithm\n");
printf("Seek Sequence: ");

// Move towards higher cylinder numbers
for (int i = index + 1; i < size; i++) {
    seek_count += abs_diff(head, temp_requests[i]);
    head = temp_requests[i];
    printf("%d ", head);
}

// Jump to the lowest request
if (index != 0) {
    seek_count += abs_diff(head, temp_requests[0]);
```

```
        head = temp_requests[0];
        printf("%d ", head);

        // Continue servicing the remaining requests
        for (int i = 1; i < index; i++) {
            seek_count += abs_diff(head, temp_requests[i]);
            head = temp_requests[i];
            printf("%d ", head);
        }
    }

    printf("\nTotal Seek Operations: %d\n", seek_count);
    printf("Average Seek Time: %.2f\n", (float)seek_count / n);
}

int main() {
    int n, head, disk_size;
    int requests[MAX_REQUESTS];

    printf("Enter the number of disk requests: ");
    scanf("%d", &n);

    printf("Enter the disk requests (cylinder numbers): ");
    for (int i = 0; i < n; i++)
        scanf("%d", &requests[i]);

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

```
    printf("Enter the total number of cylinders in the disk: ");
    scanf("%d", &disk_size);

    SSTF(requests, n, head);
    SCAN(requests, n, head, disk_size);
    C_LOOK(requests, n, head);

    return 0;
}

/*OUTPUT –
Enter the number of disk requests: 8
Enter the disk requests (cylinder numbers): 98 183 37 122 14 124 65 67
Enter the initial head position: 53
Enter the total number of cylinders in the disk: 200
*/
```