**NAME: G. GANESH**

**REG NO: 192373008**

# EXERICSE-39

**Develop a C program to simulate C-SCAN disk scheduling algorithm.**

**AIM:**

To develop a C program to simulate the C-SCAN (Circular SCAN) disk scheduling algorithm**.**

**Algorithm:**

1. Input the number of disk requests, disk size (range), and initial head position.
2. Take the direction of head movement (upwards or downwards).
3. Sort the disk requests in ascending order.
4. Divide the requests into two parts:
   - Requests less than the initial head position.
   - Requests greater than or equal to the initial head position.
5. Process the requests in the direction of head movement:
   - Service all requests in the current direction until the end or start of the disk.
   - Jump to the opposite end of the disk and continue servicing requests.
6. Calculate the total seek time.
7. Output the seek sequence and total seek time.

**Procedure:**

1. Sort the disk requests for traversal order.
2. Traverse requests in the specified direction until the end of the disk.
3. Jump to the opposite end and process remaining requests.
4. Calculate and sum the seek times.
5. Print the seek sequence and total seek time.

**Code:**

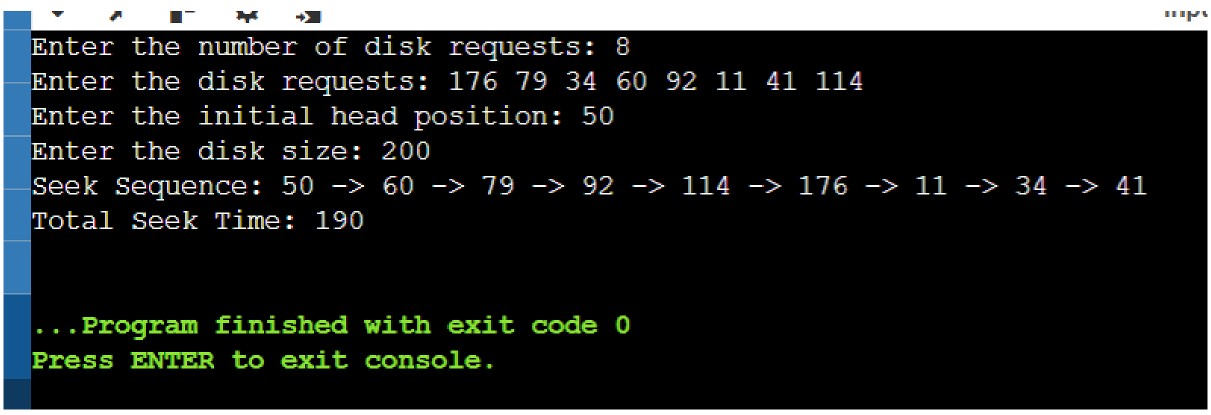
```
#include <stdio.h>

#include <stdlib.h>

void c_scan(int requests[], int n, int head, int diskSize) {

   int totalSeekTime = 0, current = head;

   int seekSequence[n + 2], index = 0;

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

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

         if (requests[j] > requests[j + 1]) {

            int temp = requests[j];
```

```
                requests[j] = requests[j + 1];
                requests[j + 1] = temp;
            }
        }
    }
    int lower[n], higher[n], lowerCount = 0, higherCount = 0;
    for (int i = 0; i < n; i++) {
        if (requests[i] < head) lower[lowerCount++] = requests[i];
        else higher[higherCount++] = requests[i];
    }
    for (int i = 0; i < higherCount; i++) {
        totalSeekTime += abs(current - higher[i]);
        seekSequence[index++] = higher[i];
        current = higher[i];
    }
    totalSeekTime += abs(current - (diskSize - 1));
    current = 0;
    for (int i = 0; i < lowerCount; i++) {
        totalSeekTime += abs(current - lower[i]);
        seekSequence[index++] = lower[i];
        current = lower[i];
    }
    printf("Seek Sequence: %d -> ", head);
    for (int i = 0; i < index; i++) {
        printf("%d", seekSequence[i]);
        if (i != index - 1) printf(" -> ");
    }
    printf("\nTotal Seek Time: %d\n", totalSeekTime);
}
int main() {
    int n, head, diskSize;
    printf("Enter the number of disk requests: ");
    scanf("%d", &n);
    int requests[n];
    printf("Enter the disk requests: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &requests[i]);
    }
    printf("Enter the initial head position: ");
    scanf("%d", &head);
    printf("Enter the disk size: ");
    scanf("%d", &diskSize);
    c_scan(requests, n, head, diskSize);
    return 0;
}
```

**Result:**

The program successfully simulates the C-SCAN disk scheduling algorithm by processing requests in a circular manner, servicing all requests in the direction of the head movement, jumping to the opposite end of the disk, and continuing. It displays the seek sequence and calculates the total seek time effectively.

**Output:**



```
Enter the number of disk requests: 8
Enter the disk requests: 176 79 34 60 92 11 41 114
Enter the initial head position: 50
Enter the disk size: 200
Seek Sequence: 50 -> 60 -> 79 -> 92 -> 114 -> 176 -> 11 -> 34 -> 41
Total Seek Time: 190


...Program finished with exit code 0
Press ENTER to exit console.
```