

D.GUNA VARDHAN  
192372041

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

#### **AIM**

To design a C program that simulates the **C-SCAN Disk Scheduling Algorithm**, where the disk arm moves in one direction to the end of the disk, then jumps to the beginning of the disk and moves in the opposite direction to service the remaining requests.

#### **ALGORITHM**

1. **Start**
2. Read the total number of disk requests and their corresponding track numbers.
3. Sort the disk track requests in increasing order.
4. Separate the requests into two groups:
  - Requests to the left of the initial head position.
  - Requests to the right of the initial head position.
5. First, move the disk head to the right to service all the requests in the right group until it reaches the end of the disk.
6. Jump to the beginning of the disk (position 0).
7. Then, move the disk head to the left to service all the requests in the left group.
8. Calculate the total number of movements made by the disk arm.
9. Print the sequence of serviced requests and the total number of disk movements.
10. **Stop**

#### **PROCEDURE**

1. Include necessary libraries (stdio.h for input/output and stdlib.h for memory management).
2. Read the total number of disk requests and their track numbers.
3. Sort the disk track numbers in increasing order to simulate the C-SCAN algorithm.

4. Separate the requests into two groups based on the initial position of the disk head (left and right).
5. Simulate the movement of the disk arm, first servicing the requests in the right direction, then jumping back to position 0 to service the remaining requests in the left direction.
6. Calculate the total number of disk movements as the sum of the absolute differences between the current position and the serviced request.
7. Display the total number of disk movements and the sequence of serviced requests.
8. **End**

CODE:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
void C_SCAN(int arr[], int n, int start, int total_tracks) {
```

```
    int total_distance = 0;
```

```
    int current_position = start;
```

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

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

```
            if (arr[i] > arr[j]) {
```

```
                int temp = arr[i];
```

```
                arr[i] = arr[j];
```

```
                arr[j] = temp;
```

```
            }
```

```
        }
```

```
    }
```

```
    int left[n], right[n];
```

```
int left_count = 0, right_count = 0;

for (int i = 0; i < n; i++) {
    if (arr[i] < start) {
        left[left_count++] = arr[i];
    } else {
        right[right_count++] = arr[i];
    }
}

int i;

for (i = 0; i < right_count; i++) {
    total_distance += abs(right[i] - current_position);
    current_position = right[i];
}

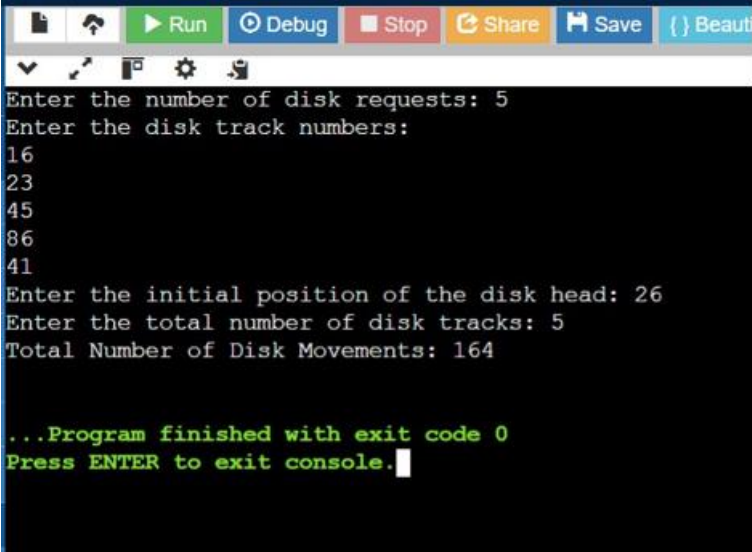
total_distance += abs(current_position - total_tracks);
current_position = 0;

for (i = 0; i < left_count; i++) {
    total_distance += abs(left[i] - current_position);
    current_position = left[i];
}

printf("Total Number of Disk Movements: %d\n", total_distance);
}
```

```
int main() {  
    int n, start, total_tracks;  
  
    printf("Enter the number of disk requests: ");  
    scanf("%d", &n);  
  
    int arr[n];  
  
    printf("Enter the disk track numbers:\n");  
    for (int i = 0; i < n; i++) {  
        scanf("%d", &arr[i]);  
    }  
  
    printf("Enter the initial position of the disk head: ");  
    scanf("%d", &start);  
  
    printf("Enter the total number of disk tracks: ");  
    scanf("%d", &total_tracks);  
  
    C_SCAN(arr, n, start, total_tracks);  
  
    return 0;  
}
```

OUTPUT:



```
Enter the number of disk requests: 5
Enter the disk track numbers:
16
23
45
86
41
Enter the initial position of the disk head: 26
Enter the total number of disk tracks: 5
Total Number of Disk Movements: 164

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