38. Design a C program to simulate SCAN disk scheduling algorithm.

AIM

To design a C program that simulates the **SCAN Disk Scheduling Algorithm**, where the disk arm moves in one direction to service requests until it reaches the end of the disk, then reverses direction and services requests in the opposite direction.

ALGORITHM

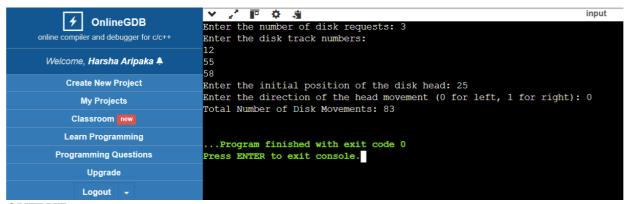
- 1. Start
- 2. Read the total number of disk requests and their corresponding track numbers.
- 3. Sort the disk track requests in increasing order.
- 4. Separate the requests into two groups:
 - o Requests to the left of the initial head position.
 - o Requests to the right of the initial head position.
- 5. If the head moves to the left, service the requests in the left group first, then reverse direction to service the requests in the right group.
- 6. If the head moves to the right, service the requests in the right group first, then reverse direction to service the requests in the left group.
- 7. Calculate the total number of movements made by the disk arm.
- 8. Print the sequence of serviced requests and the total number of disk movements.
- 9. 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 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 one direction, then reversing the direction to service the remaining requests.
- 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 SCAN(int arr[], int n, int start, int direction) {
  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;
  if (direction == 0) {
     for (i = left\_count - 1; i >= 0; i--) {
       total_distance += abs(left[i] - current_position);
       current_position = left[i];
     }
     total_distance += abs(current_position);
     current_position = 0;
     for (i = 0; i < right\_count; i++) {
       total_distance += abs(right[i] - current_position);
       current_position = right[i];
     }
  } else {
     for (i = 0; i < right\_count; i++) {
```

```
total_distance += abs(right[i] - current_position);
       current_position = right[i];
     }
     total_distance += abs(current_position);
     current_position = 0;
     for (i = left\_count - 1; i >= 0; 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, direction;
  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 direction of the head movement (0 for left, 1 for right): ");
  scanf("%d", &direction);
  SCAN(arr, n, start, direction);
  return 0;
```



OUTPUT: