B. Siva Shirish-192324016

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:
    - Requests to the left of the initial head position.
    - 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

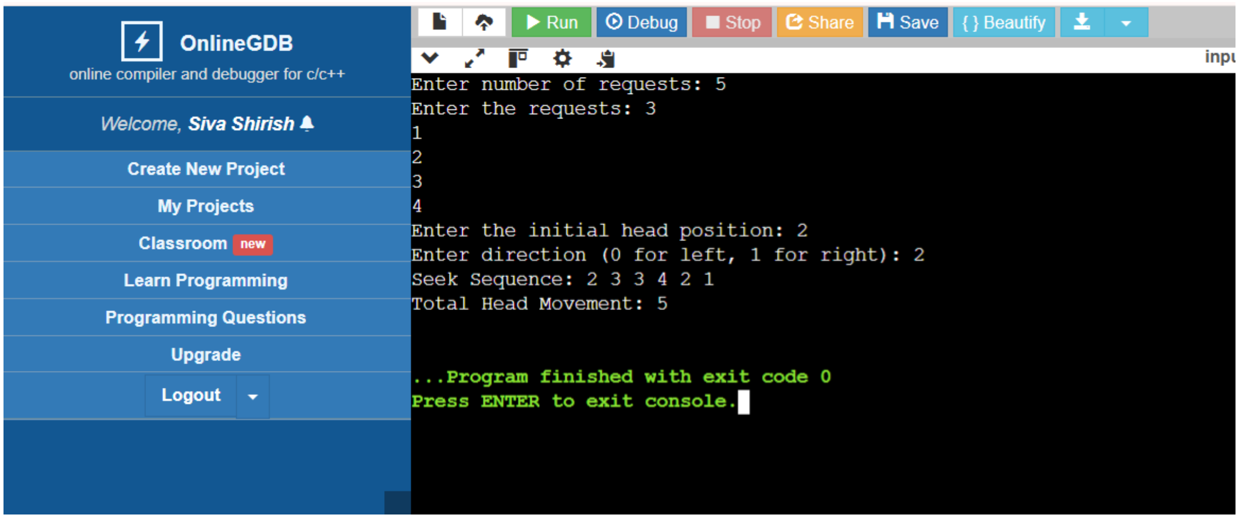
PROCEDURE

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>

#define MAX 100

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

void scan(int arr[], int n, int head, int direction) {
    int seek_sequence[MAX], index = 0, distance, total_head_movement = 0;
    int i;

    sort(arr, n);

    int start = 0, end = n - 1;

    while (start <= end && arr[start] < head) start++;
    while (start <= end && arr[end] > head) end--;

    if (direction == 0) {
        for (i = start - 1; i >= 0; i--) {
            seek_sequence[index++] = arr[i];
        }
        seek_sequence[index++] = head;
        for (i = start; i < n; i++) {
            seek_sequence[index++] = arr[i];
        }
    } else {
        for (i = start; i < n; i++) {
            seek_sequence[index++] = arr[i];
        }
        seek_sequence[index++] = head;
        for (i = start - 1; i >= 0; i--) {
            seek_sequence[index++] = arr[i];
        }
    }

    for (i = 0; i < index - 1; i++) {
        distance = abs(seek_sequence[i + 1] - seek_sequence[i]);
        total_head_movement += distance;
    }

    printf("Seek Sequence: ");
    for (i = 0; i < index; i++) {
        printf("%d ", seek_sequence[i]);
    }
    printf("\nTotal Head Movement: %d\n", total_head_movement);
}

int main() {
    int arr[MAX], n, head, direction;

    printf("Enter number of requests: ");
    scanf("%d", &n);
    printf("Enter the requests: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter the initial head position: ");
    scanf("%d", &head);
    printf("Enter direction (0 for left, 1 for right): ");
    scanf("%d", &direction);

    scan(arr, n, head, direction);

    return 0;
}
```

Output:



```
Enter number of requests: 5
Enter the requests: 3
1
2
3
4
Enter the initial head position: 2
Enter direction (0 for left, 1 for right): 2
Seek Sequence: 2 3 3 4 2 1
Total Head Movement: 5


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