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

19.cpp

```c
#include <stdio.h>
#include <math.h>
int main()
{
    int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20],
    queue2[20], temp1 = 0, temp2 = 0;
    float avg;
    printf("Enter the max range of disk\n");
    scanf("%d", &max);
    printf("Enter the initial head position\n");
    scanf("%d", &head);
    printf("Enter the size of queue request\n");
    scanf("%d", &n);
    printf("Enter the queue of disk positions to be read\n");
    for (i = 1; i <= n; i++)
    {
        scanf("%d", &temp);
        if (temp >= head)
        {
            queue1[temp1] = temp;
            temp1++;
        }
        else
        {
            queue2[temp2] = temp;
            temp2++;
        }
    }
    for (i = 0; i < temp1 - 1; i++)
    {
        for (j = i + 1; j < temp1; j++)
        {
            if (queue1[i] > queue1[j])
            {
                temp = queue1[i];
                queue1[i] = queue1[j];
                queue1[j] = temp;
            }
        }
    }
    for (i = 0; i < temp2 - 1; i++)
    {
        for (j = i + 1; j < temp2; j++)
        {
            if (queue2[i] > queue2[j])
            {
                temp = queue2[i];
                queue2[i] = queue2[j];
                queue2[j] = temp;
            }
        }
    }
    for (i = 1, j = 0; j < temp1; i++, j++)
        queue[i] = queue1[j];
    queue[i] = max;
    queue[i + 1] = 0;
    for (i = temp1 + 3, j = 0; j < temp2; i++, j++)
        queue[i] = queue2[j];
    queue[0] = head;
```

```
    for (i = 0; i < temp1 - 1; i++)
    {
        for (j = i + 1; j < temp1; j++)
        {
            if (queue1[i] > queue1[j])
            {
                temp = queue1[i];
                queue1[i] = queue1[j];
                queue1[j] = temp;
            }
        }
    }
    for (i = 0; i < temp2 - 1; i++)
    {
        for (j = i + 1; j < temp2; j++)
        {
            if (queue2[i] > queue2[j])
            {
                temp = queue2[i];
                queue2[i] = queue2[j];
                queue2[j] = temp;
            }
        }
    }
    for (i = 1, j = 0; j < temp1; i++, j++)
        queue[i] = queue1[j];
    queue[i] = max;
    queue[i + 1] = 0;
    for (i = temp1 + 3, j = 0; j < temp2; i++, j++)
        queue[i] = queue2[j];
    queue[0] = head;
    for (j = 0; j <= n + 1; j++)
    {
        diff = abs(queue[j + 1] - queue[j]);
        seek += diff;
        printf("Disk head moves from %d to %d with seek %d\n", queue[j],
        queue[j + 1], diff);
    }
    printf("Total seek time is %d\n", seek);
    avg = seek / (float)n;
    printf("Average seek time is %f\n", avg);
    return 0;
}
```

0

```
Enter the max range of disk
4
Enter the initial head position
33
Enter the size of queue request
2
Enter the queue of disk positions to be read
2
1
Disk head moves from 33 to 4 with seek 29
Disk head moves from 4 to 0 with seek 4
Disk head moves from 0 to 1 with seek 1
Disk head moves from 1 to 2 with seek 1
Total seek time is 35
Average seek time is 17.500000

--------------------------------
Process exited after 8.849 seconds with return value 0
Press any key to continue . . .
```