Program (Bankers)

```
#include <stdio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n, r;
void input();
void show();
void cal();
int main()
{
    int i, j;
    printf("Enter the no of Processes\t");
    scanf("%d", &n);
    printf("Enter the no of resources instances\t");
    scanf("%d", &r);
    printf("Enter the Max Matrix\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            scanf("%d", &max[i][j]);
        }
    }
    printf("Enter the Allocation Matrix\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            scanf("%d", &alloc[i][j]);
        }
    }
    printf("Enter the available Resources\n");
    for (j = 0; j < r; j++)
    {
        scanf("%d", &avail[j]);
    }
    show();
    cal();
    return 0;
}
void show()
{
    int i, j;
    printf("Process\t Allocation Max Available\t");
    for (i = 0; i < n; i++)
    {
        printf("\nP%d\t\t ", i);
        for (j = 0; j < r; j++)
        {
            printf("%d ", alloc[i][j]);
        }
        printf("\t");
        for (j = 0; j < r; j++)
        {
            printf("%d ", max[i][j]);
        }
        printf("\t\t");
        if (i == 0)
        {
            for (j = 0; j < r; j++)
                printf("%d ", avail[j]);
        }
    }
}
void cal()
{
    int finish[100], temp, need[100][100], flag = 1, k, c1 = 0;
    int safe[100];
    int i, j;
    for (i = 0; i < n; i++)
    {
        finish[i] = 0;
    }
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            need[i][j] = max[i][j] - alloc[i][j];
        }
    }
    printf("\n");
    while (flag)
    {
        flag = 0;
        for (i = 0; i < n; i++)
        {
            int c = 0;
            for (j = 0; j < r; j++)
            {
                if ((finish[i] == 0) && (need[i][j] <= avail[j]))
                {
                    c++;
                    if (c == r)
                    {
                        for (k = 0; k < r; k++)
                        {
                            avail[k] += alloc[i][k];
                            finish[i] = 1;
                            flag = 1;
                        }
                        printf("P%d->", i);
                    }
                }
            }
        }
    }
    for (i = 0; i < n; i++){
        if (finish[i] == 1)
        {
            c1++;
        }
        else
        {
            printf("P%d->", i);
        }
    }
    if (c1 == n){
        printf("\n The system is in safe state");
    }
    else
    {
        printf("\n Process are in dead lock");
        printf("\n System is in unsafe state");
    }
}
```

Output

```
Enter the no of Processes: 5
Enter the no of resources instances: 3
Enter the Max Matrix
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3

Enter the Allocation Matrix
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2

Enter the available Resources
3 3 2
Process  Allocation     Max         Available
P0       0 1 0          7 5 3       3 3 2
P1       2 0 0          3 2 2
P2       3 0 2          9 0 2
P3       2 1 1          2 2 2
P4       0 0 2          4 3 3

P1->P3->P4->P0->P2->
The system is in safe state
```

Program (Disk Scheduling – FCFS)

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 25
int n, head, seekCount, tracks[MAX];
void fcfsdisk()
{
    int curr_track, distance;
    seekCount = 0;
    for (int i = 0; i < n; i++)
    {
        curr_track = tracks[i];
        distance = abs(head - curr_track);
        seekCount += distance;
        head = curr_track;
    }
}
int main()
{
    printf("\n FCFS Disk Scheduling\n");
    printf("\n Enter the number of tracks to be seeked : ");
    scanf("%d", &n);
    if (n > MAX)
    {
        printf("\n Number of tracks to be seeked cannot exceed %d. Exiting...\
n", MAX);
        exit(0);
    }
    printf("\n Enter the starting position of the head : ");
    scanf("%d", &head);
    printf("\n Enter the tracks to be seeked : ");
    for (int i = 0; i < n; i++)
        scanf("%d", &tracks[i]);
    fcfsdisk();
    printf("\n The Seek Sequence is : ");
    for (int i = 0; i < n - 1; i++)
        printf(" %d -> ", tracks[i]);
    printf(" %d\n", tracks[n - 1]);
    printf("\n The Seek Count is : %d\n", seekCount);
    return (0);
}
```

Output

```
FCFS Disk Scheduling

Enter the number of tracks to be seeked : 3

Enter the starting position of the head : 5

Enter the tracks to be seeked : 2 9 4

The Seek Sequence is :  2 ->  9 ->  4

The Seek Count is : 15
```

Program (Disk Scheduling – SCAN)

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 25
int n, head, size, seekCount, tracks[MAX], sequence[MAX];
char dir;
void sort(int arr[], int m)
{
    int temp;
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < m - 1 - i; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
void scandisk()
{
    int currTrack, distance, l = 0, r = 0, left[MAX], right[MAX];
    seekCount = 0;
    if (dir == 'L')
    {
        left[0] = 0;
        l++;
    }
    else if (dir == 'R')
    {
        right[0] = size - 1;
        r++;
    }
    for (int i = 0; i < n; i++)
    {
        if (tracks[i] < head)
            left[l++] = tracks[i];
        if (tracks[i] > head)
            right[r++] = tracks[i];
    }
    sort(left, l);
    sort(right, r);
    int run = 2, x = 0;
    while (run-- > 0)
    {
        if (dir == 'L')
        {
            for (int i = l - 1; i >= 0; i--)
            {
                currTrack = left[i];
                sequence[x++] = currTrack;
                distance = abs(head - currTrack);
                seekCount += distance;
                head = currTrack;
            }
            dir = 'R';
        }
        else
        {
            for (int i = 0; i < r; i++)
            {
                currTrack = right[i];
                sequence[x++] = currTrack;
                distance = abs(head - currTrack);
                seekCount += distance;
                head = currTrack;
            }
            dir = 'L';
        }
    }
}
int main()
{
    int i;
    printf("\n SCAN Disk Scheduling\n");
    printf("\n Enter the size of the disk : ");
    scanf("%d", &size);
    printf("\n Enter the number of tracks to be seeked : ");
    scanf("%d", &n);
    if (n > MAX)
    {
        printf("\n Number of tracks to be seeked cannot exceed %d Exiting...\n",
MAX);
        exit(0);
    }
    printf("\n Enter the starting position of the head : ");
    scanf("%d", &head);
    if (head > size)
    {
        printf("\n Starting position of head cannot exceed the size of disk.
Exiting...\n");
        exit(0);
    }
    printf("\n Enter the initial direction of the head(L/R) : ");
    scanf(" %c", &dir);
    if ((dir != 'L') && (dir != 'R'))
    {
        printf("\n Invalid direction input. Exiting...\n");
        exit(0);
    }
    printf("\n Enter the tracks to be seeked : ");
    for (int i = 0; i < n; i++)
        scanf("%d", &tracks[i]);
    scandisk();
    printf("\n The Seek Sequence is : ");
    for (i = 0; i < n; i++)
        printf(" %d -> ", sequence[i]);
    printf(" %d\n", sequence[i]);
    printf("\n The Seek Count is : %d\n", seekCount);
    return 0;
}
```

Output

```
SCAN Disk Scheduling
Enter the size of the disk : 10
Enter the number of tracks to be seeked : 3
Enter the starting position of the head : 5
Enter the initial direction of the head(L/R) : R
Enter the tracks to be seeked : 2 6 9
The Seek Sequence is :  6 ->  9 ->  9 ->  2
The Seek Count is : 11
```

Program (disk scheduling c-scan)

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 25
int n, head, size, seekCount, tracks[MAX], sequence[MAX];
void sort(int arr[], int m)
{
    int temp;
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < m - 1 - i; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
void cscandisk()
{
    int currTrack, distance, l, r, left[MAX], right[MAX];
    seekCount = 0;
    l = 0;
    r = 0;
    left[0] = 0;
    l++;
    right[0] = size - 1;
    r++;
    for (int i = 0; i < n; i++)
    {
        if (tracks[i] < head)
            left[l++] = tracks[i];
        if (tracks[i] > head)
            right[r++] = tracks[i];
    }
    sort(left, l);
    sort(right, r);
    int x = 0;
    for (int i = 0; i < r; i++)
    {
        currTrack = right[i];
        sequence[x++] = currTrack;
        distance = abs(head - currTrack);
        seekCount += distance;
        head = currTrack;
    }
    head = 0;
    seekCount += size - 1;
    for (int i = 0; i < l; i++)
    {
        currTrack = left[i];
        sequence[x++] = currTrack;
        distance = abs(head - currTrack);
        seekCount += distance;
        head = currTrack;
    }
}
int main()
{
    int i;
    printf("\n C-SCAN Disk Scheduling\n");
    printf("\n Enter the size of the disk : ");
    scanf("%d", &size);
    printf("\n Enter the number of tracks to be seeked : ");
    scanf("%d", &n);
    if (n > MAX)
    {
        printf("\n Number of tracks to be seeked cannot exceed %d Exiting...\n",
MAX);
        exit(0);
    }
    printf("\n Enter the starting position of the head : ");
    scanf("%d", &head);
    if (head > size)
    {
        printf("\n Starting position of head cannot exceed the size of disk.
Exiting...\n");
        exit(0);
    }
    printf("\n Enter the tracks to be seeked : ");
    for (int i = 0; i < n; i++)
        scanf("%d", &tracks[i]);
    cscandisk();
    printf("\n The Seek Sequence is : ");
    for (i = 0; i < n; i++)
        printf(" %d -> ", sequence[i]);
    printf(" %d\n", sequence[i]);
    printf("\n The Seek Count is : %d\n", seekCount);
    return 0;
}
```

<u>Output</u>

```
C-SCAN Disk Scheduling

Enter the size of the disk : 10
Enter the number of tracks to be seeked : 3
Enter the starting position of the head : 5
Enter the tracks to be seeked : 5 7 4
The Seek Sequence is :  7 ->  9 ->  0 ->  4
The Seek Count is : 17
```