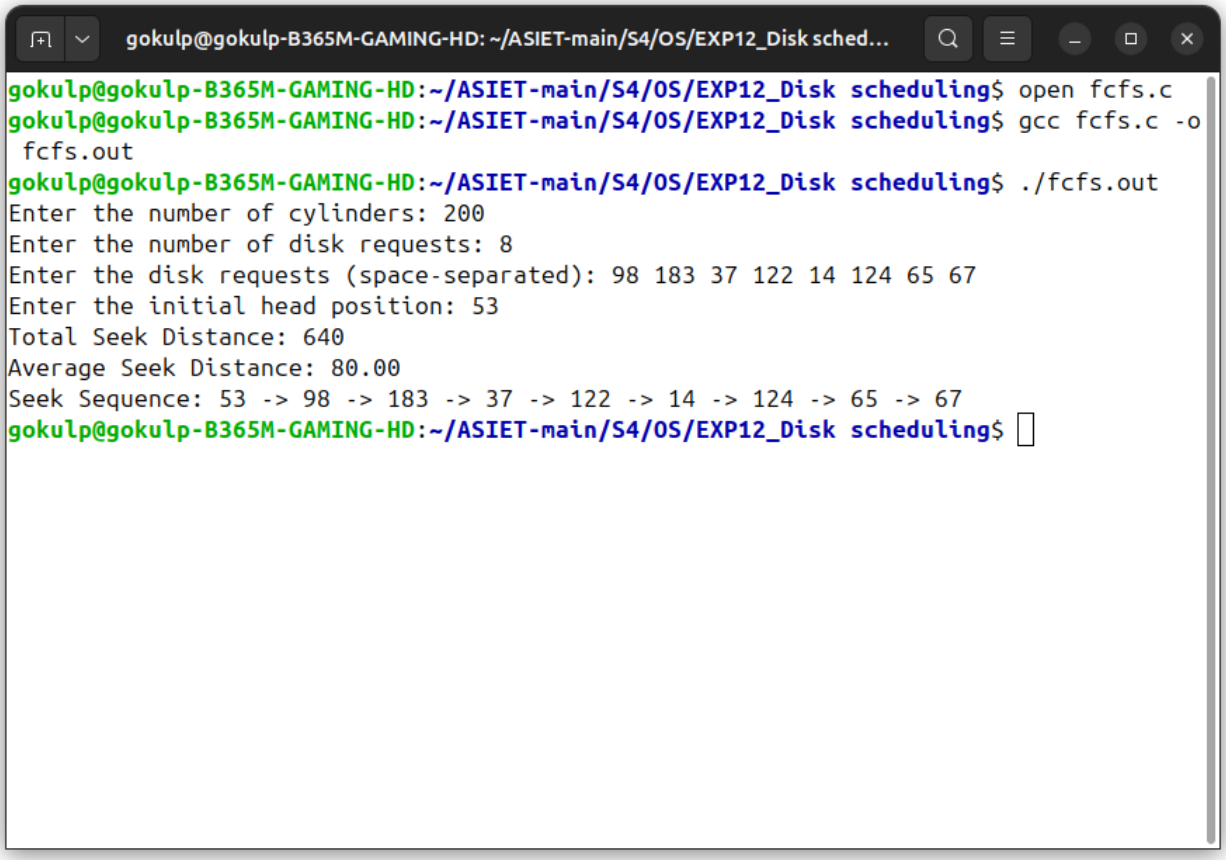
First Come First Serve (FCFS)

Program

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

void printSeekSequence(int sequence[], int n, int init) {
   int i;
   printf("Seek Sequence: ");
   printf("%d -> ",init);
   for (i = 0; i < n - 1; i++) {
      printf("%d -> ", sequence[i]);
   }
   printf("%d\n", sequence[n - 1]);
}

void fcfs(int tracks[], int n, int initial, int totalCylinders) {
   int totalSeekDistance = 0, i;
   int seekSequence[n];
   int current = initial;
   seekSequence[0] = current;

   for (i = 0; i < n; i++) {
      if (tracks[i] < 0 || tracks[i] >= totalCylinders) {
         printf("Error: Disk request %d is out of bounds (valid range: 0 to %d).\n", tracks[i],
totalCylinders - 1);
         return;
      }
   }

   for (i = 0; i < n; i++) {
      totalSeekDistance += abs(current - tracks[i]);
      current = tracks[i];
      seekSequence[i] = current;
   }

   float avgSeekDistance = (float)totalSeekDistance / n;

   printf("Total Seek Distance: %d\n", totalSeekDistance);
   printf("Average Seek Distance: %.2f\n", avgSeekDistance);
   printSeekSequence(seekSequence, n,initial);
}
```

```
int main() {
    int n, initial, totalCylinders, i;

    printf("Enter the number of cylinders: ");
    scanf("%d", &totalCylinders);

    printf("Enter the number of disk requests: ");
    scanf("%d", &n);

    int tracks[n];

    printf("Enter the disk requests (space-separated): ");
    for (i = 0; i < n; i++) {
        scanf("%d", &tracks[i]);
    }

    printf("Enter the initial head position: ");
    scanf("%d", &initial);

    if (initial < 0 || initial >= totalCylinders) {
        printf("Error: Initial head position is out of bounds (valid range: 0 to %d).\n", totalCylinders -
1);
        return 1;
    }

    fcfs(tracks, n, initial, totalCylinders);
    return 0;
}
```

Output


gokulp@gokulp-B365M-GAMING-HD: ~/ASIET-main/S4/OS/EXP12_Disk sched...
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ open fcfs.c
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ gcc fcfs.c -o
fcfs.out
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ ./fcfs.out
Enter the number of cylinders: 200
Enter the number of disk requests: 8
Enter the disk requests (space-separated): 98 183 37 122 14 124 65 67
Enter the initial head position: 53
Total Seek Distance: 640
Average Seek Distance: 80.00
Seek Sequence: 53 -> 98 -> 183 -> 37 -> 122 -> 14 -> 124 -> 65 -> 67
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$

SCAN (ELEVATOR)

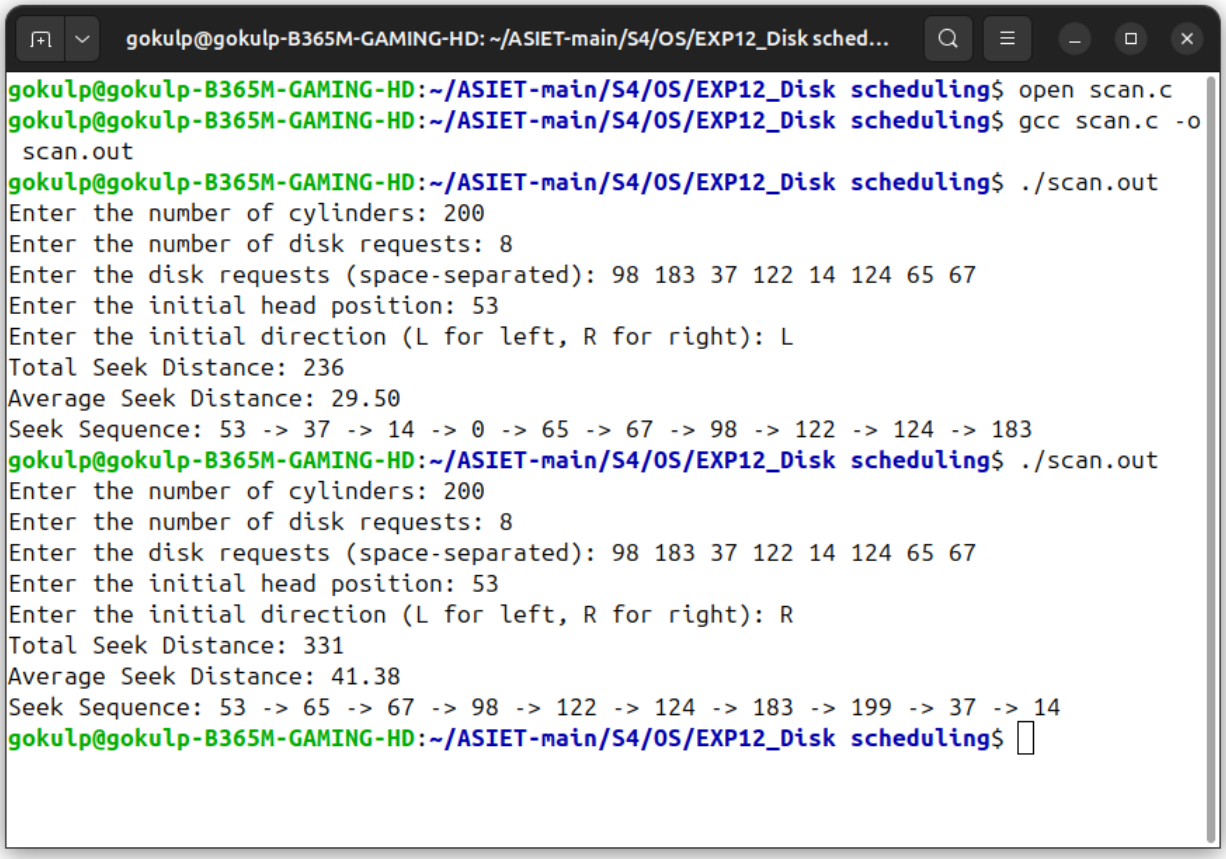
Program

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

void printSeekSequence(int sequence[], int n, int init) {
    printf("Seek Sequence: %d", init);
    for (int i = 0; i < n; i++) {
        printf(" -> %d", sequence[i]);
    }
    printf("\n");
}

void scan(int tracks[], int n, int initial, int totalCylinders, char direction) {
    int totalSeekDistance = 0;
    int seekSequence[n + 2];
    int current = initial;
    int count = 0;

    for (int i = 0; i < n; i++) {
        if (tracks[i] < 0 || tracks[i] >= totalCylinders) {
            printf("Error: Disk request %d is out of bounds (valid range: 0 to %d).\n", tracks[i],
totalCylinders - 1);
            return;
        }
    }

    // Sorting the request array
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (tracks[i] > tracks[j]) {
                int temp = tracks[i];
                tracks[i] = tracks[j];
                tracks[j] = temp;
            }
        }
    }

    int left[n], right[n], leftCount = 0, rightCount = 0;

    for (int i = 0; i < n; i++) {
        if (tracks[i] < initial) {
            left[leftCount++] = tracks[i];
        } else {
            right[rightCount++] = tracks[i];
        }
    }

    int seek = 0;
    if (direction == 'R' || direction == 'r') {
```

```
        for (int i = 0; i < rightCount; i++) {
            seekSequence[count++] = right[i];
            seek += abs(current - right[i]);
            current = right[i];
        }
        if (rightCount > 0 && leftCount > 0) {
            seekSequence[count++] = totalCylinders - 1;
            seek += abs(current - (totalCylinders - 1));
            current = totalCylinders - 1;
        }
        for (int i = leftCount - 1; i >= 0; i--) {
            seekSequence[count++] = left[i];
            seek += abs(current - left[i]);
            current = left[i];
        }
    }
    else if (direction == 'L' || direction == 'l') {
        for (int i = leftCount - 1; i >= 0; i--) {
            seekSequence[count++] = left[i];
            seek += abs(current - left[i]);
            current = left[i];
        }
        if (leftCount > 0 && rightCount > 0) {
            seekSequence[count++] = 0;
            seek += abs(current - 0);
            current = 0;
        }
        for (int i = 0; i < rightCount; i++) {
            seekSequence[count++] = right[i];
            seek += abs(current - right[i]);
            current = right[i];
        }
    } else {
        printf("Error: Invalid direction input. Please enter 'L' for left or 'R' for right.\n");
        return;
    }

    printf("Total Seek Distance: %d\n", seek);
    printf("Average Seek Distance: %.2f\n", (float)seek / n);
    printSeekSequence(seekSequence, count, initial);
}
```

```
int main() {
    int n, initial, totalCylinders;
    char direction;

    printf("Enter the number of cylinders: ");
    scanf("%d", &totalCylinders);

    printf("Enter the number of disk requests: ");
    scanf("%d", &n);

    int tracks[n];
    printf("Enter the disk requests (space-separated): ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &tracks[i]);
    }

    printf("Enter the initial head position: ");
    scanf("%d", &initial);
    if (initial < 0 || initial >= totalCylinders) {
        printf("Error: Initial head position is out of bounds (valid range: 0 to %d).\n", totalCylinders -
1);
        return 1;
    }

    printf("Enter the initial direction (L for left, R for right): ");
    scanf(" %c", &direction);

    scan(tracks, n, initial, totalCylinders, direction);
    return 0;
}
```

Output


gokulp@gokulp-B365M-GAMING-HD: ~/ASIET-main/S4/OS/EXP12_Disk sched...
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ open scan.c
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ gcc scan.c -o scan.out
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ ./scan.out
Enter the number of cylinders: 200
Enter the number of disk requests: 8
Enter the disk requests (space-separated): 98 183 37 122 14 124 65 67
Enter the initial head position: 53
Enter the initial direction (L for left, R for right): L
Total Seek Distance: 236
Average Seek Distance: 29.50
Seek Sequence: 53 -> 37 -> 14 -> 0 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ ./scan.out
Enter the number of cylinders: 200
Enter the number of disk requests: 8
Enter the disk requests (space-separated): 98 183 37 122 14 124 65 67
Enter the initial head position: 53
Enter the initial direction (L for left, R for right): R
Total Seek Distance: 331
Average Seek Distance: 41.38
Seek Sequence: 53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 199 -> 37 -> 14
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$

Circular SCAN (Circular Elevator)

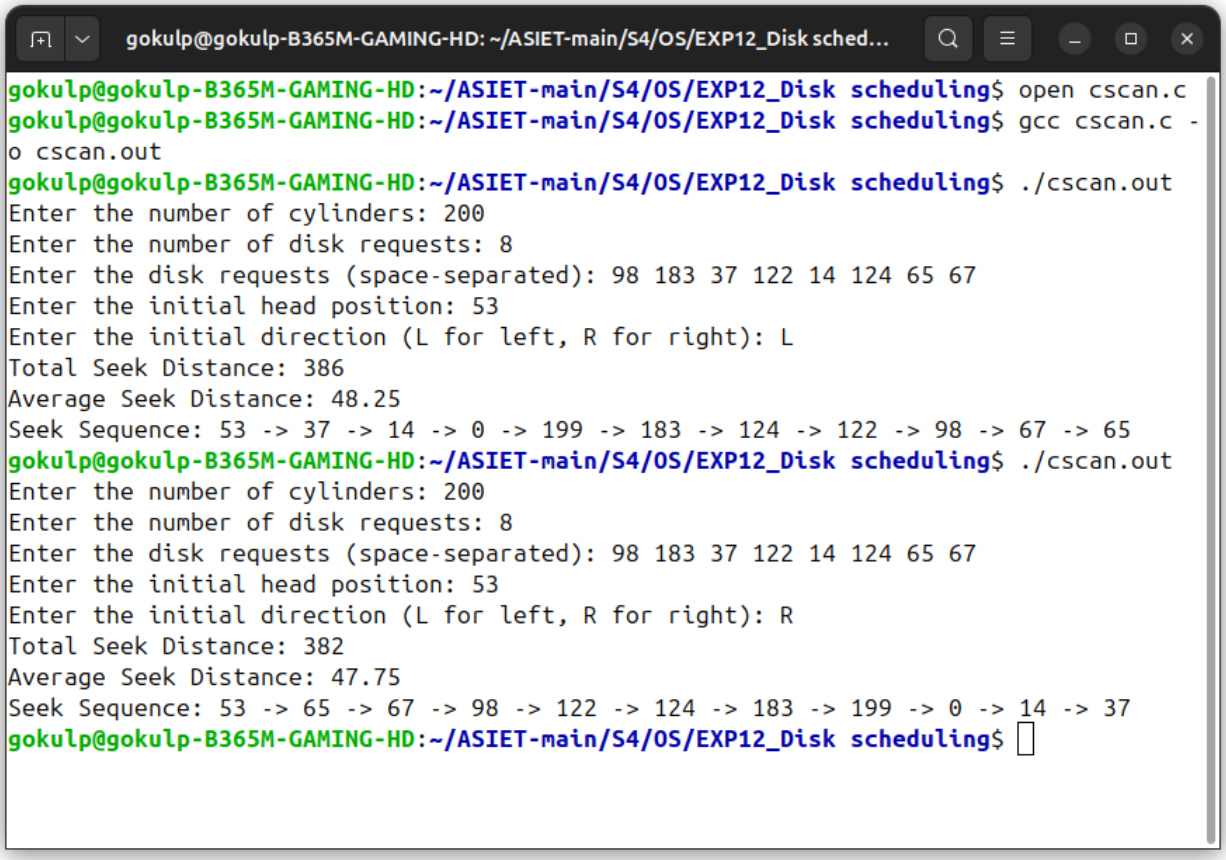
Program

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

void printSeekSequence(int sequence[], int n, int init) {
    printf("Seek Sequence: %d", init);
    for (int i = 0; i < n; i++) {
        printf(" -> %d", sequence[i]);
    }
    printf("\n");
}

void cscan(int tracks[], int n, int initial, int totalCylinders, char direction) {
    int totalSeekDistance = 0;
    int seekSequence[n + 2];
    int current = initial;
    int count = 0;

    for (int i = 0; i < n; i++) {
        if (tracks[i] < 0 || tracks[i] >= totalCylinders) {
            printf("Error: Disk request %d is out of bounds (valid range: 0 to %d).\n", tracks[i],
totalCylinders - 1);
            return;
        }
    }

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

    int left[n], right[n], leftCount = 0, rightCount = 0;
```

```
for (int i = 0; i < n; i++) {
      if (tracks[i] < initial) {
         left[leftCount++] = tracks[i];
      } else {
         right[rightCount++] = tracks[i];
      }
  }

  int seek = 0;
  if (direction == 'R' || direction == 'r') {
      for (int i = 0; i < rightCount; i++) {
         seekSequence[count++] = right[i];
         seek += abs(current - right[i]);
         current = right[i];
      }
      if (rightCount > 0) {
         seekSequence[count++] = totalCylinders - 1;
         seek += abs(current - (totalCylinders - 1));
         current = 0;
         seekSequence[count++] = 0;
         seek += totalCylinders - 1; // Add the jump distance
      }
      for (int i = 0; i < leftCount; i++) {
         seekSequence[count++] = left[i];
         seek += abs(current - left[i]);
         current = left[i];
      }
  }
  else if (direction == 'L' || direction == 'l') {
      for (int i = leftCount - 1; i >= 0; i--) {
         seekSequence[count++] = left[i];
         seek += abs(current - left[i]);
         current = left[i];
      }
      if (leftCount > 0) {
         seekSequence[count++] = 0;
         seek += abs(current - 0);
         current = totalCylinders - 1;
         seekSequence[count++] = totalCylinders - 1;
         seek += totalCylinders - 1; // Add the jump distance
      }
```

```
        for (int i = rightCount - 1; i >= 0; i--) {
            seekSequence[count++] = right[i];
            seek += abs(current - right[i]);
            current = right[i];
        }
    } else {
        printf("Error: Invalid direction input. Please enter 'L' for left or 'R' for right.\n");
        return;
    }

    printf("Total Seek Distance: %d\n", seek);
    printf("Average Seek Distance: %.2f\n", (float)seek / n);
    printSeekSequence(seekSequence, count, initial);
}

int main() {
    int n, initial, totalCylinders;
    char direction;

    printf("Enter the number of cylinders: ");
    scanf("%d", &totalCylinders);

    printf("Enter the number of disk requests: ");
    scanf("%d", &n);

    int tracks[n];
    printf("Enter the disk requests (space-separated): ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &tracks[i]);
    }

    printf("Enter the initial head position: ");
    scanf("%d", &initial);
    if (initial < 0 || initial >= totalCylinders) {
        printf("Error: Initial head position is out of bounds (valid range: 0 to %d).\n", totalCylinders -
1);
        return 1;
    }

    printf("Enter the initial direction (L for left, R for right): ");
    scanf(" %c", &direction);

    cscan(tracks, n, initial, totalCylinders, direction);
    return 0;
}
```

OUTPUT


gokulp@gokulp-B365M-GAMING-HD: ~/ASIET-main/S4/OS/EXP12_Disk sched...
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ open cscan.c
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ gcc cscan.c -o cscan.out
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ ./cscan.out
Enter the number of cylinders: 200
Enter the number of disk requests: 8
Enter the disk requests (space-separated): 98 183 37 122 14 124 65 67
Enter the initial head position: 53
Enter the initial direction (L for left, R for right): L
Total Seek Distance: 386
Average Seek Distance: 48.25
Seek Sequence: 53 -> 37 -> 14 -> 0 -> 199 -> 183 -> 124 -> 122 -> 98 -> 67 -> 65
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$ ./cscan.out
Enter the number of cylinders: 200
Enter the number of disk requests: 8
Enter the disk requests (space-separated): 98 183 37 122 14 124 65 67
Enter the initial head position: 53
Enter the initial direction (L for left, R for right): R
Total Seek Distance: 382
Average Seek Distance: 47.75
Seek Sequence: 53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 199 -> 0 -> 14 -> 37
gokulp@gokulp-B365M-GAMING-HD:~/ASIET-main/S4/OS/EXP12_Disk scheduling$