```
//FIFO Disk Scheduling Algorithm :

#include <stdio.h>
#include <stdlib.h>
int main() {
int n, i, head, total_movement = 0;
printf("Enter number of disk requests: ");
scanf("%d", &n);
int requests[n];
printf("Enter the request sequence: ");
for (i = 0; i < n; i++) {
scanf("%d", &requests[i]);
}
printf("Enter initial head position: ");
scanf("%d", &head);
printf("\nDisk Sequence: %d", head);
for (i = 0; i < n; i++) {
total_movement += abs(requests[i] - head);
head = requests[i];
printf("  %d", head);
}
printf("\n\nTotal Head Movement: %d", total_movement);
printf("\nAverage Seek Time: %.2f\n", (float)total_movement / n);
return 0;
}
```

```
C - LOOK Disk Scheduling Algorithm :


#include <stdio.h>
#include <stdlib.h>
int main() {
int n, i, j, head, total_movement = 0, direction, index;
printf("Enter number of disk requests: ");
scanf("%d", &n);
int requests[n];
printf("Enter the request sequence: ");
for (i = 0; i < n; i++)
scanf("%d", &requests[i]);
printf("Enter initial head position: ");
scanf("%d", &head);
printf("Enter head movement direction (1 for high/right, 0 for low/left): ");
scanf("%d", &direction);
// Sort requests
for (i = 0; i < n - 1; i++) {
for (j = i + 1; j < n; j++) {
if (requests[i] > requests[j]) {
int temp = requests[i];
requests[i] = requests[j];
requests[j] = temp;
}
}
}
// Find index of first request greater than head
for (i = 0; i < n; i++) {
if (head < requests[i]) {
index = i;
break;
}
}
printf("\nSeek Sequence: %d", head);
// Move towards higher tracks
if (direction == 1) {
for (i = index; i < n; i++) {
total_movement += abs(head - requests[i]);
head = requests[i];
printf("  %d", head);
}
// Jump to the lowest request
total_movement += abs(head - requests[0]);
head = requests[0];
printf("  %d", head);
for (i = 1; i < index; i++) {
total_movement += abs(head - requests[i]);
head = requests[i];
printf("  %d", head);
}
}
// Move towards lower tracks
else {
for (i = index - 1; i >= 0; i--) {
total_movement += abs(head - requests[i]);
head = requests[i];
printf("  %d", head);
}
// Jump to the highest request
total_movement += abs(head - requests[n - 1]);
head = requests[n - 1];
printf("  %d", head);
for (i = n - 2; i >= index; i--) {
total_movement += abs(head - requests[i]);
head = requests[i];
printf("  %d", head);
}
}
printf("\n\nTotal Head Movement: %d", total_movement);
printf("\nAverage Head Movement: %.2f\n", (float)total_movement / n);
return 0;
}
```