

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 void sort(int arr[], int n) {
6     for (int i = 0; i < n; i++) {
7         for (int j = i + 1; j < n; j++) {
8             if (arr[i] > arr[j]) {
9                 int temp = arr[i];
10                arr[i] = arr[j];
11                arr[j] = temp;
12            }
13        }
14    }
15 }
16
17
18 int findClosestRequest(int requests[], int n, int head, int visited[]) {
19     int minDiff = 10000; // Large value to find the minimum difference
20     int closestIdx = -1;
21
22     for (int i = 0; i < n; i++) {
23         if (!visited[i]) {
24             int diff = abs(requests[i] - head);
25             if (diff < minDiff) {
26                 minDiff = diff;
27                 closestIdx = i;
28             }
29         }
30     }
31
32     return closestIdx;
33 }
34
35 void SSTF(int requests[], int n, int head) {
36     int visited[n];
37     for (int i = 0; i < n; i++) {
38         visited[i] = 0;
39     }
40
41     int seekCount = 0;
42     int sequence[n];
43
44     printf("Seek sequence (SSTF): ");
45     for (int i = 0; i < n; i++) {
46         int closestIdx = findClosestRequest(requests, n, head, visited);
47         visited[closestIdx] = 1;
48         seekCount += abs(requests[closestIdx] - head);
49         head = requests[closestIdx];
50         sequence[i] = head;
51         printf("%d ", head);

```

```

52     }
53
54     printf("\nTotal Seek Time (SSTF): %d\n", seekCount);
55 }
56
57 // SCAN (Elevator Algorithm)
58 void SCAN(int requests[], int n, int head, int disk_size, int direction) {
59     int seekCount = 0;
60     int sequence[n + 2];
61     int idx = 0;
62
63     sort(requests, n);
64
65     printf("Seek sequence (SCAN): ");
66
67     // Service requests in the specified direction
68     if (direction == 1) { // Move towards higher
69         for (int i = 0; i < n; i++) {
70             if (requests[i] > head) {
71                 sequence[idx++] = requests[i];
72                 seekCount += abs(requests[i] - head);
73                 head = requests[i];
74             }
75         }
76         // Go to the end of the disk
77         sequence[idx++] = disk_size;
78         seekCount += abs(disk_size - head);
79         head = disk_size;
80
81         // Move back in the opposite direction
82         for (int i = n - 1; i >= 0; i--) {
83             if (requests[i] < head) {
84                 sequence[idx++] = requests[i];
85                 seekCount += abs(requests[i] - head);
86                 head = requests[i];
87             }
88         }
89     } else { // Move towards lower
90         for (int i = n - 1; i >= 0; i--) {
91             if (requests[i] < head) {
92                 sequence[idx++] = requests[i];
93                 seekCount += abs(requests[i] - head);
94                 head = requests[i];
95             }
96         }
97         // Go to the start of the disk (0)
98         sequence[idx++] = 0;
99         seekCount += head; // Head was at the last serviced request
100
101         // Move in the other direction
102         for (int i = 0; i < n; i++) {
103             if (requests[i] > head) {

```

```

104         sequence[idx++] = requests[i];
105         seekCount += abs(requests[i] - head);
106         head = requests[i];
107     }
108 }
109 }
110
111 // Print the seek sequence
112 for (int i = 0; i < idx; i++) {
113     printf("%d ", sequence[i]);
114 }
115
116 printf("\nTotal Seek Time (SCAN): %d\n", seekCount);
117 }
118
119 // C-LOOK (Circular LOOK)
120 void C_LOOK(int requests[], int n, int head) {
121     int seekCount = 0;
122     int sequence[n + 1];
123     int idx = 0;
124
125     sort(requests, n);
126
127     printf("Seek sequence (C-LOOK): ");
128
129     // Service requests to the right of the head
130     for (int i = 0; i < n; i++) {
131         if (requests[i] > head) {
132             sequence[idx++] = requests[i];
133             seekCount += abs(requests[i] - head);
134             head = requests[i];
135         }
136     }
137
138     // Jump back to the beginning of the requests (smallest)
139     seekCount += abs(requests[0] - head);
140     head = requests[0];
141     sequence[idx++] = head;
142
143     // Service the remaining requests
144     for (int i = 1; i < n; i++) {
145         if (requests[i] > head) {
146             sequence[idx++] = requests[i];
147             seekCount += abs(requests[i] - head);
148             head = requests[i];
149         }
150     }
151
152     // Print the seek sequence
153     for (int i = 0; i < idx; i++) {
154         printf("%d ", sequence[i]);
155     }

```

```

156
157     printf("\nTotal Seek Time (C-LOOK): %d\n", seekCount);
158 }
159
160 // Menu-driven program
161 int main() {
162     int n, head, disk_size, direction, choice;
163
164     printf("Enter the number of disk requests: ");
165     scanf("%d", &n);
166
167     int requests[n];
168     printf("Enter the disk request queue: ");
169     for (int i = 0; i < n; i++) {
170         scanf("%d", &requests[i]);
171     }
172
173     printf("Enter the initial head position: ");
174     scanf("%d", &head);
175
176     while (1) {
177         printf("\nDisk Scheduling Algorithms:\n");
178         printf("1. SSTF\n");
179         printf("2. SCAN\n");
180         printf("3. C-LOOK\n");
181         printf("4. Exit\n");
182         printf("Enter your choice: ");
183         scanf("%d", &choice);
184
185         switch (choice) {
186             case 1:
187                 SSTF(requests, n, head);
188                 break;
189             case 2:
190                 printf("Enter disk size (max cylinder): ");
191                 scanf("%d", &disk_size);
192                 printf("Enter initial direction (1 for high, 0 for low): ");
193                 scanf("%d", &direction);
194                 SCAN(requests, n, head, disk_size, direction);
195                 break;
196             case 3:
197                 C_LOOK(requests, n, head);
198                 break;
199             case 4:
200                 exit(0);
201             default:
202                 printf("Invalid choice! Please select again.\n");
203         }
204     }
205
206     return 0;
207 }

```

