Assignment:-8

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

int abs(int n) {
   return n < 0 ? -n : n;
}

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

void fcfs(int data[], int n, int head) {
   int movement = 0;
   int current = head;
   for (int i = 0; i < n; i++) {
      movement += abs(current - data[i]);
      current = data[i];
   }
   printf("Total Head Movement (FCFS): %d\n", movement);
}

void sstf(int data[], int n, int head) {
   int done[100] = {0}, movement = 0, count = 0, current = head;
   while (count < n) {
      int index = -1, min = 99999;
      for (int i = 0; i < n; i++) {
         if (!done[i] && abs(current - data[i]) < min) {
            min = abs(current - data[i]);
            index = i;
         }
      }
      movement += min;
      current = data[index];
      done[index] = 1;
      count++;
   }
   printf("Total Head Movement (SSTF): %d\n", movement);
}
```

```
void scan(int data[], int n, int head, int direction, int x, int y) {
    int movement = 0, i, current = head;
    sort(data, n);

    printf("Seek Sequence: %d", current);
    if (direction == 1) {
        for (i = 0; i < n; i++) {
            if (data[i] >= current) {
                movement += abs(current - data[i]);
                current = data[i];
                printf(" -> %d", current);
            }
        }
        if (current != y) {
            movement += abs(current - y);
            current = y;
        }
        for (i = n - 1; i >= 0; i--) {
            if (data[i] < head) {
                movement += abs(current - data[i]);
                current = data[i];
                printf(" -> %d", current);
            }
        }
    } else {
        for (i = n - 1; i >= 0; i--) {
            if (data[i] <= current) {
                movement += abs(current - data[i]);
                current = data[i];
                printf(" -> %d", current);
            }
        }
        if (current != x) {
            movement += abs(current - x);
            current = x;
        }
        for (i = 0; i < n; i++) {
            if (data[i] > head) {
                movement += abs(current - data[i]);
                current = data[i];
                printf(" -> %d", current);
            }
        }
    }
    printf("\nTotal Head Movement (SCAN): %d\n", movement);
}

void cscan(int data[], int n, int head, int direction, int x, int y) {
    int movement = 0, i, current = head;
    sort(data, n);
```

```
    if (direction == 1) {
        for (i = 0; i < n; i++) {
            if (data[i] >= current) {
                movement += abs(current - data[i]);
                current = data[i];
            }
        }
        if (current != y) movement += abs(current - y);
        current = x;
        movement += abs(y - x);
        for (i = 0; i < n; i++) {
            if (data[i] < head) {
                movement += abs(current - data[i]);
                current = data[i];
            }
        }
    } else {
        for (i = n - 1; i >= 0; i--) {
            if (data[i] <= current) {
                movement += abs(current - data[i]);
                current = data[i];
            }
        }
        if (current != x) movement += abs(current - x);
        current = y;
        movement += abs(y - x);
        for (i = n - 1; i >= 0; i--) {
            if (data[i] > head) {
                movement += abs(current - data[i]);
                current = data[i];
            }
        }
    }

    printf("Total Head Movement (C-SCAN): %d\n", movement);
}

int main() {
    int data[100], n, head, x, y, direction, choice;
    printf("Enter number of disk requests: ");
    scanf("%d", &n);
    printf("Enter disk request queue: ");
    for (int i = 0; i < n; i++) scanf("%d", &data[i]);
    printf("Enter initial head position: ");
    scanf("%d", &head);

    printf("\nChoose Disk Scheduling Algorithm:\n");
    printf("1. FCFS\n2. SSTF\n3. SCAN\n4. C-SCAN\nChoice: ");
    scanf("%d", &choice);
```

```
    if (choice == 3 || choice == 4) {
        printf("Enter direction (1 for right, 0 for left): ");
        scanf("%d", &direction);
        printf("Enter disk range (min and max):\n");
        printf("Min: ");
        scanf("%d", &x);
        printf("Max: ");
        scanf("%d", &y);
    }

    switch (choice) {
        case 1:
            fcfs(data, n, head);
            break;
        case 2:
            sstf(data, n, head);
            break;
        case 3:
            scan(data, n, head, direction, x, y);
            break;
        case 4:
            cscan(data, n, head, direction, x, y);
            break;
        default:
            printf("Invalid choice.\n");
    }

    return 0;
}
```

OUTPUT

```
Choose Disk Scheduling Algorithm:
1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
Choice: 2
Total Head Movement (SSTF): 279
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft
t-MIEngine-Out-vfekmg20.gzz"
@pravinmahato ➜/workspaces/Algorithm (main) $
```

```
Choose Disk Scheduling Algorithm:
1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
Choice: 3
Enter direction (1 for right, 0 for left): 1
Enter disk range (min and max):
Min: 0
Max: 199
Seek Sequence: 53 -> 53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 198 -> 37 -> 14 -> 0
Total Head Movement (SCAN): 345
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-
t-MIEngine-Out-d5twgxkc.5jx"
@pravinmahato ➜/workspaces/Algorithm (main) $
```

```
Choose Disk Scheduling Algorithm:
1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
Choice: 4
Enter direction (1 for right, 0 for left): 1
Enter disk range (min and max):
Min: 0
Max: 199
Total Head Movement (C-SCAN): 382
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft
t-MIEngine-Out-oatxlxg4.unn"
@pravinmahato ➜/workspaces/Algorithm (main) $
```