Assignment No 6:

Name: Samruddhi Devram Khilari

Branch & Year: AIML-B SY

Prn: 12420020

Roll no: 2

Sub: Operating System Lab

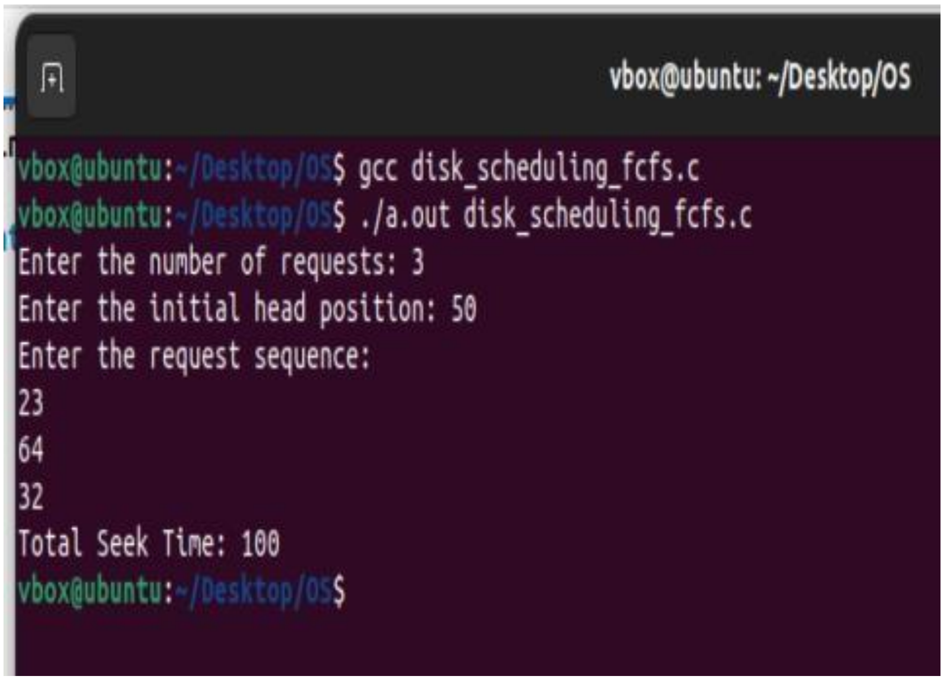
Aim: - Disk scheduling Technique

Q1] (FCFS)

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

int

main()    {

int   n,   i,

head,

seekTime

= 0;

   // Accept the number of

requests     printf("Enter the

number of requests: ");

scanf("%d", &n);

   int requests[n];

   // Accept the initial head

position     printf("Enter the

initial head position: ");

scanf("%d", &head);

   // Accept the sequence of

requests     printf("Enter the

request sequence:\n");

   for (i = 0; i <

n;      i++)      {
```

```
scanf("%d",

&requests[i]);

    }

    // Calculate total seek time for FCFS
    for (i = 0; i < n; i++) {        seekTime += abs(requests[i] - head);  // Calculate seek
time from current head position to the request        head = requests[i];                  //
Update head to the current request position
// Output total seek time

printf("Total Seek Time: %d\n",

seekTime);

    return 0;
}
```

Output ===



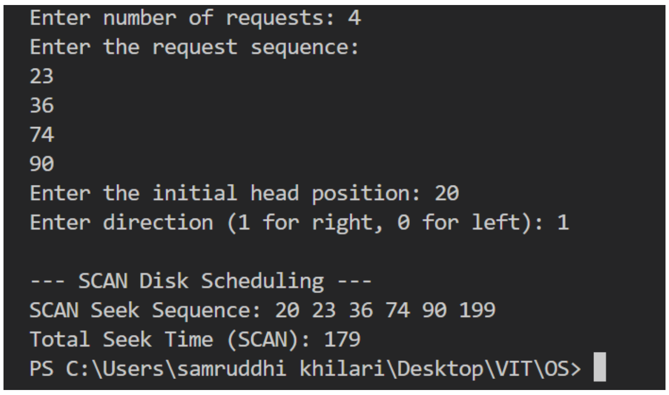
```
vbox@ubuntu: ~/Desktop/OS
vbox@ubuntu:~/Desktop/OS$ gcc disk_scheduling_fcfs.c
vbox@ubuntu:~/Desktop/OS$ ./a.out disk_scheduling_fcfs.c
Enter the number of requests: 3
Enter the initial head position: 50
Enter the request sequence:
23
64
32
Total Seek Time: 100
vbox@ubuntu:~/Desktop/OS$
```

Q2] SCAN

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

#define MAX_TRACKS 200 // Assuming a maximum of 200 tracks

// Function to implement SCAN Disk Scheduling
void scan(int requests[], int numRequests, int head, int direction) {
  int seekSequence[MAX_TRACKS];
  int distance, totalSeek = 0, index = 0;

  // Create a temporary array to store sorted requests
  int temp[MAX_TRACKS], tempIndex = 0;
  for (int i = 0; i < numRequests; i++) {
    temp[tempIndex++] = requests[i];
  }

  // Sort requests
  for (int i = 0; i < tempIndex; i++) {
    for (int j = i + 1; j < tempIndex; j++) {
      if (temp[i] > temp[j]) {
        int tempVal = temp[i];
        temp[i] = temp[j];
        temp[j] = tempVal;
      }
    }
  }

  // Add the head position to the array
  int totalTracks = 0;
  if (direction == 1) { // Moving right
    for (int i = 0; i < tempIndex; i++) {
      if (temp[i] >= head) {
        seekSequence[totalTracks++] = head;
        for (int j = i; j < tempIndex; j++) {
          seekSequence[totalTracks++] = temp[j];
        }
        seekSequence[totalTracks++] = MAX_TRACKS - 1; // Go to the end
        for (int j = i - 1; j >= 0; j--) {
          seekSequence[totalTracks++] = temp[j];
        }
        break;
      }
    }
  } else { // Moving left
    for (int i = tempIndex - 1; i >= 0; i--) {
      if (temp[i] <= head) {
        seekSequence[totalTracks++] = head;
        for (int j = i; j >= 0; j--) {
          seekSequence[totalTracks++] = temp[j];
        }
        seekSequence[totalTracks++] = 0; // Go to the beginning
```

```
                for (int j = i + 1; j < tempIndex; j++) {
                    seekSequence[totalTracks++] = temp[j];
                }
                break;
            }
        }
    }

    // Calculate total seek time
    for (int i = 0; i < totalTracks - 1; i++) {
        distance = abs(seekSequence[i + 1] - seekSequence[i]);
        totalSeek += distance;
    }

    printf("SCAN Seek Sequence: ");
    for (int i = 0; i < totalTracks; i++) {
        printf("%d ", seekSequence[i]);
    }
    printf("\nTotal Seek Time (SCAN): %d\n", totalSeek);
}

int main() {
    int requests[MAX_TRACKS], numRequests, head, direction;

    printf("Enter number of requests: ");
    scanf("%d", &numRequests);
    printf("Enter the request sequence:\n");
    for (int i = 0; i < numRequests; i++) {
        scanf("%d", &requests[i]);
    }
    printf("Enter the initial head position: ");
    scanf("%d", &head);
    printf("Enter direction (1 for right, 0 for left): ");
    scanf("%d", &direction);

    printf("\n--- SCAN Disk Scheduling ---\n");
    scan(requests, numRequests, head, direction);

    return 0;
}
```

Output==

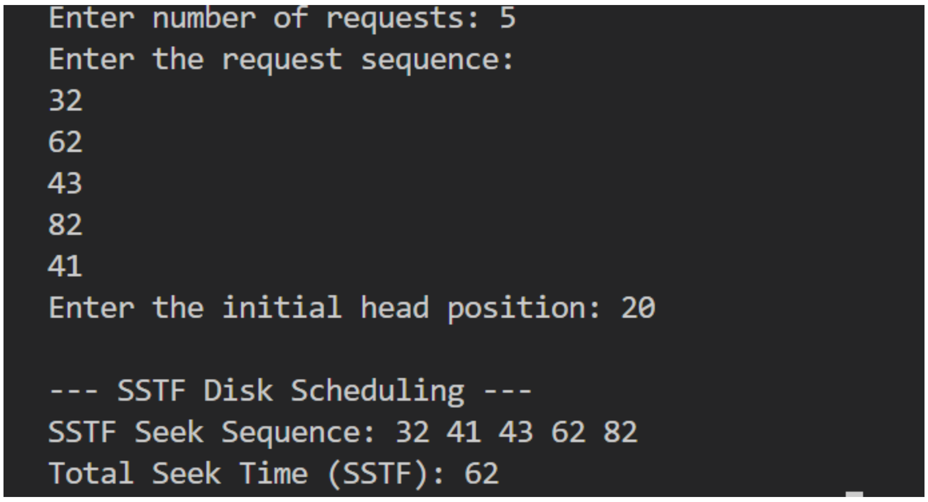


```
Enter number of requests: 4
Enter the request sequence:
23
36
74
90
Enter the initial head position: 20
Enter direction (1 for right, 0 for left): 1

--- SCAN Disk Scheduling ---
SCAN Seek Sequence: 20 23 36 74 90 199
Total Seek Time (SCAN): 179
PS C:\Users\samruddhi khilari\Desktop\VIT\OS>
```

Q3] Shortest Seek Time First

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

#define MAX_TRACKS 200 // Assuming a maximum of 200 tracks

// Function to implement SSTF Disk Scheduling
void sstf(int requests[], int numRequests, int head) {
    int seekSequence[MAX_TRACKS];
    int visited[MAX_TRACKS] = {0}; // To keep track of visited requests
    int totalSeek = 0;

    int currentHead = head;
    int totalTracks = 0;

    for (int i = 0; i < numRequests; i++) {
        int minDistance = 1000; // Initialize to a large value
        int minIndex = -1;

        // Find the closest request
        for (int j = 0; j < numRequests; j++) {
            if (!visited[j]) { // If request is not yet visited
                int distance = abs(requests[j] - currentHead);
                if (distance < minDistance) {
                    minDistance = distance;
                    minIndex = j;
                }
            }
        }

        // Visit the closest request
        visited[minIndex] = 1;
        seekSequence[totalTracks++] = requests[minIndex];
        totalSeek += minDistance;
        currentHead = requests[minIndex];
    }

    // Print the seek sequence and total seek time
    printf("SSTF Seek Sequence: ");
    for (int i = 0; i < totalTracks; i++) {
        printf("%d ", seekSequence[i]);
    }
    printf("\nTotal Seek Time (SSTF): %d\n", totalSeek);
}

int main() {
    int requests[MAX_TRACKS], numRequests, head;

    printf("Enter number of requests: ");
    scanf("%d", &numRequests);
    printf("Enter the request sequence:\n");
    for (int i = 0; i < numRequests; i++) {
        scanf("%d", &requests[i]);
    }
    printf("Enter the initial head position: ");
    scanf("%d", &head);

    printf("\n--- SSTF Disk Scheduling ---\n");
    sstf(requests, numRequests, head);

    return 0;
}
```

Output ===

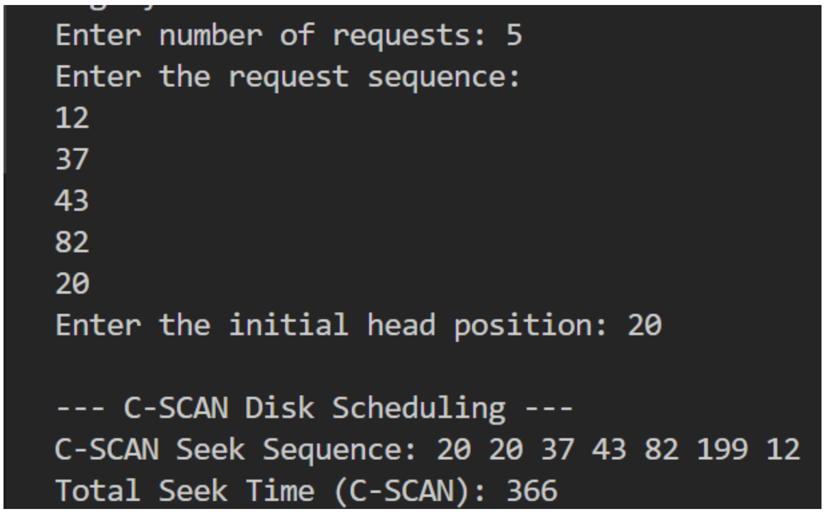


```
Enter number of requests: 5
Enter the request sequence:
32
62
43
82
41
Enter the initial head position: 20

--- SSTF Disk Scheduling ---
SSTF Seek Sequence: 32 41 43 62 82
Total Seek Time (SSTF): 62
```

Q4] C-SCAN

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

#define MAX_TRACKS 200 // Assuming a maximum of 200 tracks

// Function to implement C-SCAN Disk Scheduling
void cscan(int requests[], int numRequests, int head) {
    int seekSequence[MAX_TRACKS];
    int distance, totalSeek = 0, index = 0;

    // Create a temporary array to store sorted requests
    int temp[MAX_TRACKS], tempIndex = 0;
    for (int i = 0; i < numRequests; i++) {
        temp[tempIndex++] = requests[i];
    }

    // Sort requests
    for (int i = 0; i < tempIndex; i++) {
        for (int j = i + 1; j < tempIndex; j++) {
            if (temp[i] > temp[j]) {
                int tempVal = temp[i];
                temp[i] = temp[j];
                temp[j] = tempVal;
            }
        }
    }
    // Add the head position to the array
    int totalTracks = 0;
    seekSequence[totalTracks++] = head;

    // Move to the right first
    for (int i = 0; i < tempIndex; i++) {
        if (temp[i] >= head) {
            seekSequence[totalTracks++] = temp[i];
        }
    }

    // Jump to the end
    seekSequence[totalTracks++] = MAX_TRACKS - 1; // Go to the end

    // Continue from the beginning to the head
    for (int i = 0; i < tempIndex; i++) {
        if (temp[i] < head) {
            seekSequence[totalTracks++] = temp[i];
        }
    }

    // Calculate total seek time
    for (int i = 0; i < totalTracks - 1; i++) {
        distance = abs(seekSequence[i + 1] - seekSequence[i]);
        totalSeek += distance;
    }

    printf("C-SCAN Seek Sequence: ");
    for (int i = 0; i < totalTracks; i++) {
        printf("%d ", seekSequence[i]);
    }
    printf("\nTotal Seek Time (C-SCAN): %d\n", totalSeek);
}

int main() {
    int requests[MAX_TRACKS], numRequests, head;

    printf("Enter number of requests: ");
    scanf("%d", &numRequests);
    printf("Enter the request sequence:\n");
    for (int i = 0; i < numRequests; i++) {
        scanf("%d", &requests[i]);
    }
    printf("Enter the initial head position: ");
    scanf("%d", &head);

    printf("\n--- C-SCAN Disk Scheduling ---\n");
    cscan(requests, numRequests, head);

    return 0;
}
```

Output ===



```
Enter number of requests: 5
Enter the request sequence:
12
37
43
82
20
Enter the initial head position: 20

--- C-SCAN Disk Scheduling ---
C-SCAN Seek Sequence: 20 20 37 43 82 199 12
Total Seek Time (C-SCAN): 366
```