Name : Bramha Nimbalkar

Roll No : 7

Div : E

ASSIGNMENT 11

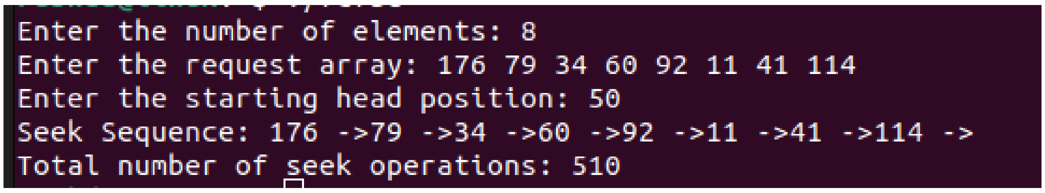
Problem : Implement in C language following disk scheduling algorithm

First Come First Serve (FCFS) disk scheduling algorithm

Code:

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

#define MAX_SIZE 100

void FCFS(int arr[], int size, int head) {
   int seek_count = 0;
   int cur_track, distance;

   for (int i = 0; i < size; i++) {
      cur_track = arr[i];

      distance = fabs(head - cur_track);

      seek_count += distance;

      head = cur_track;
   }

   printf("Total number of seek operations: %d\n", seek_count);

   printf("Seek Sequence is\n");

   for (int i = 0; i < size; i++) {
      printf("%d ->", arr[i]);
   }
}
```

```
int main() {
    int size;

    printf("Enter the number of elements: ");
    scanf("%d", &size);

    int arr[MAX_SIZE];

    printf("Enter the request array: ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    int head;

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


    FCFS(arr, size, head);

    return 0;
}
```



```
Enter the number of elements: 8
Enter the request array: 176 79 34 60 92 11 41 114
Enter the starting head position: 50
Seek Sequence: 176 ->79 ->34 ->60 ->92 ->11 ->41 ->114 ->
Total number of seek operations: 510
```

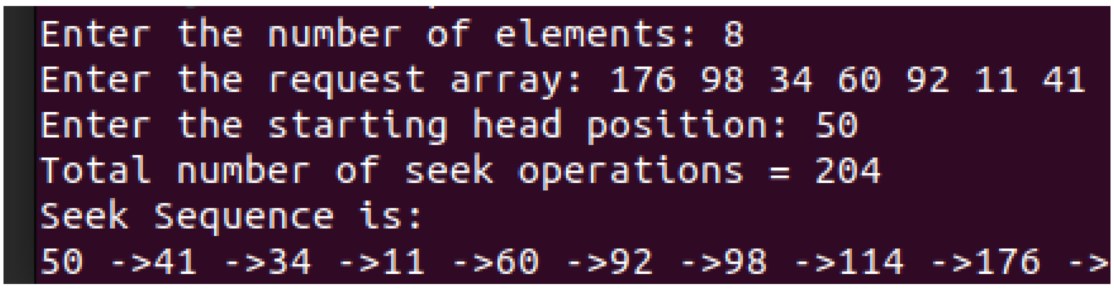
2. Shortest Seek Time First (SSTF) disk scheduling algorithm

Code

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

void calculateDifference(int queue[], int head, int diff[][2], int n) {
    for (int i = 0; i < n; i++) {
        diff[i][0] = abs(queue[i] - head);
    }
}
```

```
int findMin(int diff[][2], int n) {
   int index = -1;
   int minimum = 999999999;

   for (int i = 0; i < n; i++) {
      if (!diff[i][1] && minimum > diff[i][0]) {
         minimum = diff[i][0];
         index = i;
      }
   }
   return index;
}

void shortestSeekTimeFirst(int request[], int head, int n) {
   if (n == 0) {
      return;
   }

   int diff[n][2];
   for (int i = 0; i < n; i++) {
      diff[i][0] = 0;
      diff[i][1] = 0;
   }

   int seekCount = 0;
   int seekSequence[n + 1];

   for (int i = 0; i < n; i++) {
      seekSequence[i] = head;
      calculateDifference(request, head, diff, n);
      int index = findMin(diff, n);
      diff[index][1] = 1;

      seekCount += diff[index][0];
      head = request[index];
   }
   seekSequence[n] = head;

   printf("Total number of seek operations = %d\n", seekCount);
   printf("Seek Sequence is:\n");

   for (int i = 0; i <= n; i++) {
      printf("%d ->", seekSequence[i]);
   }
}

int main() {
   int n;
```

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

    int proc[n];

    printf("Enter the request array: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &proc[i]);
    }

    int head;

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

    shortestSeekTimeFirst(proc, head, n);

    return 0;
}
```



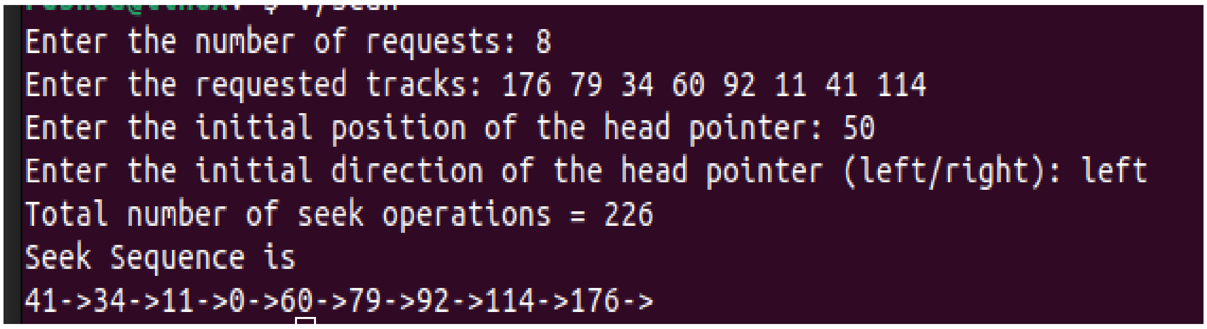
```
Enter the number of elements: 8
Enter the request array: 176 98 34 60 92 11 41
Enter the starting head position: 50
Total number of seek operations = 204
Seek Sequence is:
50 ->41 ->34 ->11 ->60 ->92 ->98 ->114 ->176 ->
```

3. SCAN disk scheduling algorithm

Code

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

void SCAN(int arr[], int N, int head, char* direction) {
    int seek_count = 0;
    int distance, cur_track = 0;
    int left[N], right[N];
    int leftIndex = 0, rightIndex = 0;
    int seek_sequence[2 * N];
    int sequenceIndex = 0;
```

```
if (direction[0] == 'l') {
    left[leftIndex++] = 0;
} else if (direction[0] == 'r') {
    right[rightIndex++] = 200 - 1;
}

for (int i = 0; i < N; i++) {
    if (arr[i] < head) {
        left[leftIndex++] = arr[i];
    }
    if (arr[i] > head) {
        right[rightIndex++] = arr[i];
    }
}

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

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

int run = 2;
while (run != 0) {
    if (direction[0] == 'l') {
        for (int i = leftIndex - 1; i >= 0; i--) {
            cur_track = left[i];

            seek_sequence[sequenceIndex++] = cur_track;

            // Calculate absolute distance
            distance = abs(cur_track - head);

            seek_count += distance;

            head = cur_track;
            }
            direction[0] = 'r';
        } else if (direction[0] == 'r') {
            for (int i = 0; i < rightIndex; i++) {
                cur_track = right[i];

                seek_sequence[sequenceIndex++] = cur_track;

                distance = abs(cur_track - head);


                seek_count += distance;

                head = cur_track;
            }
            direction[0] = 'l';
        }
        run--;
    }

    printf("Total number of seek operations = %d\n", seek_count);
    printf("Seek Sequence is\n");

    for (int i = 0; i < sequenceIndex; i++) {
        printf("%d->", seek_sequence[i]);
    }printf("\n");
}

int main() {
    int N;

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

    int arr[N];

    printf("Enter the requested tracks: ");
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    int head;
    char direction[10];

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

    printf("Enter the initial direction of the head pointer (left/right): ");
    scanf("%s", direction);
    SCAN(arr, N, head, direction);

    return 0;
}
```

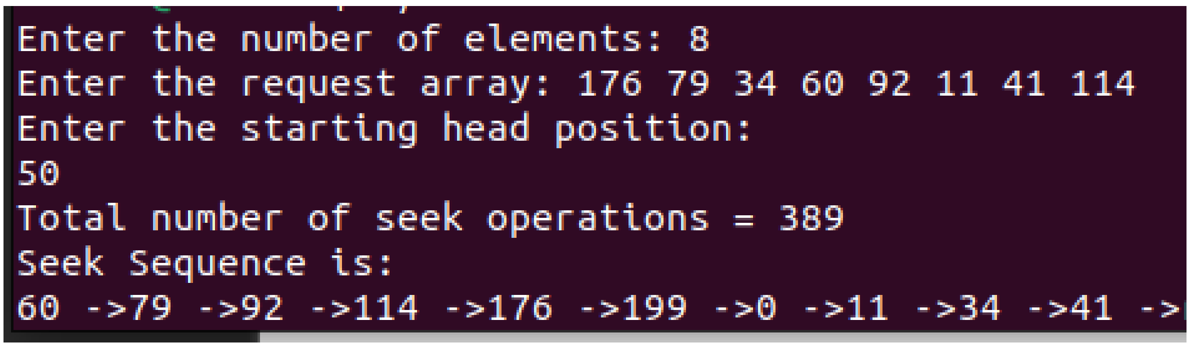


```
Enter the number of requests: 8
Enter the requested tracks: 176 79 34 60 92 11 41 114
Enter the initial position of the head pointer: 50
Enter the initial direction of the head pointer (left/right): left
Total number of seek operations = 226
Seek Sequence is
41->34->11->0->60->79->92->114->176->
```

4. Circular SCAN (C-SCAN) disk scheduling algorithm

Code

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

#define SIZE 8
#define DISK_SIZE 200

void CSCAN(int arr[], int n, int head) {
    int seekCount = 0;
    int distance = 0;
    int curTrack = 0;
    int left[SIZE];
    int right[SIZE];
    int seekSequence[2 * SIZE];

    left[0] = 0;
    right[0] = DISK_SIZE - 1;

    int leftSize = 1;
    int rightSize = 1;

    for (int i = 0; i < n; i++) {
        if (arr[i] < head) {
            left[leftSize++] = arr[i];
        }
        if (arr[i] > head) {
            right[rightSize++] = arr[i];
        }
    }

    for (int i = 0; i < leftSize - 1; i++) {
        for (int j = 0; j < leftSize - i - 1; j++) {
            if (left[j] > left[j + 1]) {
                // swap
                int temp = left[j];
                left[j] = left[j + 1];
                left[j + 1] = temp;
            }
        }
    }

    for (int i = 0; i < rightSize - 1; i++) {
        for (int j = 0; j < rightSize - i - 1; j++) {
            if (right[j] > right[j + 1]) {
                // swap
                int temp = right[j];
                right[j] = right[j + 1];
                right[j + 1] = temp;
            }
        }
    }

    for (int i = 0; i < rightSize; i++) {
        curTrack = right[i];

        seekSequence[i] = curTrack;

        distance = abs(curTrack - head);

        seekCount += distance;

        head = curTrack;
    }

    head = 0;

    seekCount += (DISK_SIZE - 1);

    for (int i = 0; i < leftSize; i++) {
        curTrack = left[i];

        seekSequence[rightSize + i] = curTrack;

        distance = abs(curTrack - head);
        seekCount += distance;

        head = curTrack;
    }

    printf("Total number of seek operations = %d\n", seekCount);
    printf("Seek Sequence is:\n");

    for (int i = 0; i < rightSize + leftSize; i++) {
        printf("%d ->", seekSequence[i]);
    }
}

int main() {
    int n;

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

    int arr[SIZE];

    printf("Enter the request array: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int head;

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

    CSCAN(arr, n, head);

    return 0;
}
```



```
Enter the number of elements: 8
Enter the request array: 176 79 34 60 92 11 41 114
Enter the starting head position:
50
Total number of seek operations = 389
Seek Sequence is:
60 ->79 ->92 ->114 ->176 ->199 ->0 ->11 ->34 ->41 ->
```