Assignment 8

8b - SCAN

Code

8b.c

```
/*
Problem Statement - To implement C programs for Disk scheduling algorithms: 2.
SCAN
*/

#include <stdio.h>

int absoluteValue(int); // Declaring function absoluteValue

void main() {
    int queue[25], n, headposition, i, j, k, seek = 0, maxrange,
        difference, temp, queue1[20], queue2[20], temp1 = 0, temp2 = 0;
    float averageSeekTime;
    // Reading the maximum Range of the Disk.
    printf("Enter the maximum range of Disk: ");
    scanf("%d", &maxrange);
    // Reading the number of Queue Requests(Disk access requests)
    printf("Enter the number of queue requests: ");
    scanf("%d", &n);
    // Reading the initial head position.(ie. the starting point of execution)
    printf("Enter the initial head position: ");
    scanf("%d", &headposition);
    // Reading disk positions to be read in the order of arrival
```

```
    printf("Enter the disk positions to be read(queue): ");
    for (i = 1;i <= n;i++) // Note that i varies from 1 to n instead of 0 to n-1
    {
        scanf("%d", &temp); //Reading position value to a temporary variable
        //Now if the requested position is greater than current headposition,
        //then pushing that to array queue1
        if (temp > headposition) {
            queue1[temp1] = temp; //temp1 is the index variable of queue1[]
            temp1++; //incrementing temp1
        }
        else //else if temp < current headposition,then push to array queue2[]
        {
            queue2[temp2] = temp; //temp2 is the index variable of queue2[]
            temp2++;
        }
    }
    //Now we have to sort the two arrays
    //SORTING array queue1[] in ascending order
    for (i = 0;i < temp1 - 1;i++) {
        for (j = i + 1;j < temp1;j++) {
            if (queue1[i] > queue1[j]) {
                temp = queue1[i];
                queue1[i] = queue1[j];
                queue1[j] = temp;
            }
        }
    }
    //SORTING array queue2[] in descending order
    for (i = 0;i < temp2 - 1;i++) {
        for (j = i + 1;j < temp2;j++) {
            if (queue2[i] < queue2[j]) {
                temp = queue2[i];
                queue2[i] = queue2[j];
                queue2[j] = temp;
            }
        }
    }
    //Copying first array queue1[] into queue[]
    for (i = 1, j = 0;j < temp1;i++, j++) {
        queue[i] = queue1[j];
    }
    //Setting queue[i] to maxrange because the head goes to
    //end of disk and comes back in scan Algorithm
    queue[i] = maxrange;
    //Copying second array queue2[] after that first one is copied, into queue[]
    for (i = temp1 + 2, j = 0;j < temp2;i++, j++) {
        queue[i] = queue2[j];
    }
    //Setting queue[i] to 0. Because that is the innermost cylinder.
    queue[i] = 0;
```

```
    //At this point, we have the queue[] with the requests in the
    //correct order of execution as per scan algorithm.
    //Now we have to set 0th index of queue[] to be the initial headposition.
    queue[0] = headposition;
    // Calculating SEEK TIME. seek is initially set to 0 in the declaration part.
    for (j = 0; j <= n; j++) //Loop starts from headposition. (ie. 0th index of
queue)
    {
        // Finding the difference between next position and current position.
        difference = absoluteValue(queue[j + 1] - queue[j]);
        // Adding difference to the current seek time value
        seek = seek + difference;
        // Displaying a message to show the movement of disk head
        printf("Disk head moves from position %d to %d with Seek %d \n", queue[j],
            queue[j + 1], difference);
    }
    // Calculating Average Seek time
    averageSeekTime = seek / (float)n;
    //Display Total and Average Seek Time(s)
    printf("Total Seek Time= %d\n", seek);
    printf("Average Seek Time= %f\n", averageSeekTime);
}
// Defining function absoluteValue
int absoluteValue(int x) {
    if (x > 0) {
        return x;
    }
    else {
        return x * -1;
    }
}
```

Output

```
abhishek-jadhav@abhishek-jadhav-ubuntu:~/Codes/OS Assignments/33232$ ./a.out
Enter the maximum range of Disk: 199
Enter the number of queue requests: 5
Enter the initial head position: 30
Enter the disk positions to be read(queue): 40
20
160
134
11
Disk head moves from position 30 to 40 with Seek 10
Disk head moves from position 40 to 134 with Seek 94
Disk head moves from position 134 to 160 with Seek 26
```

```
Disk head moves from position 160 to 199 with Seek 39
Disk head moves from position 199 to 20 with Seek 179
Disk head moves from position 20 to 11 with Seek 9
Total Seek Time= 357
Average Seek Time= 71.400002
```

8c - CLOOK

Code

8c.c

```c
/*
Problem Statement - To implement C programs for Disk scheduling algorithms: 3. C-LOOK
*/

#include <stdio.h>

int absoluteValue(int); // Declaring function absoluteValue

void main() {
    int queue[25], n, headposition, i, j, k, seek = 0, maxrange, difference, temp, queue1[20], queue2[20], temp1 = 0, temp2 = 0;
    float averageSeekTime;
    // Reading the maximum Range of the Disk.
    printf("Enter the maximum range of Disk: ");
    scanf("%d", &maxrange);
    // Reading the number of Queue Requests(Disk access requests)
    printf("Enter the number of queue requests: ");
    scanf("%d", &n);
    // Reading the initial head position.(ie. the starting point of execution)
    printf("Enter the initial head position: ");
    scanf("%d", &headposition);
    // Reading disk positions to be read in the order of arrival
    printf("Enter the disk positions to be read(queue): ");
    for (i = 1;i <= n;i++) // Note that i varies from 1 to n instead of 0 to n-1
    {
        scanf("%d", &temp); //Reading position value to a temporary variable
        //Now if the requested position is greater than current headposition,
        //then pushing that to array queue1
        if (temp > headposition) {
            queue1[temp1] = temp; //temp1 is the index variable of queue1[]
            temp1++; //incrementing temp1
```

```
            }
            else //else if temp < current headposition,then push to array queue2[]
            {
                queue2[temp2] = temp; //temp2 is the index variable of queue2[]
                temp2++;
            }
        }
        //Now we have to sort the two arrays
        //SORTING array queue1[] in ascending order
        for (i = 0;i < temp1 - 1;i++) {
            for (j = i + 1;j < temp1;j++) {
                if (queue1[i] > queue1[j]) {
                    temp = queue1[i];
                    queue1[i] = queue1[j];
                    queue1[j] = temp;
                }
            }
        }
        //SORTING array queue2[] in ascending order
        for (i = 0;i < temp2 - 1;i++) {
            for (j = i + 1;j < temp2;j++) {
                if (queue2[i] > queue2[j]) {
                    temp = queue2[i];
                    queue2[i] = queue2[j];
                    queue2[j] = temp;
                }
            }
        }
        //Copying first array queue1[] into queue[]
        for (i = 1, j = 0;j < temp1;i++, j++) {
            queue[i] = queue1[j];
        }
        //Moving Disk head to the inner most requested cylinder,
        //because this is Circular LOOK.
        queue[i] = queue2[0];
        //Copying second array queue2[] after that first one is copied, into queue[]
        for (i = temp1 + 1, j = 0;j < temp2;i++, j++) {
            queue[i] = queue2[j];
        }
        //At this point, we have the queue[] with the requests in the
        //correct order of execution as per C-LOOK algorithm.
        //Now we have to set 0th index of queue[] to be the initial headposition.
        queue[0] = headposition;
        // Calculating SEEK TIME. seek is initially set to 0 in the declaration part.
        for (j = 0; j < n; j++) //Loop starts from headposition. (ie. 0th index of
    queue)
        {
            // Finding the difference between next position and current position.
            difference = absoluteValue(queue[j + 1] - queue[j]);
            // Adding difference to the current seek time value
```

```
            seek = seek + difference;
            // Displaying a message to show the movement of disk head
            printf("Disk head moves from position %d to %d with Seek %d \n",
                queue[j], queue[j + 1], difference);
        }
        // Calculating Average Seek time
        averageSeekTime = seek / (float)n;
        //Display Total and Average Seek Time(s)
        printf("Total Seek Time= %d\n", seek);
        printf("Average Seek Time= %f\n", averageSeekTime);
    }
    // Defining function absoluteValue
    int absoluteValue(int x) {
        if (x > 0) {
            return x;
        }
        else {
            return x * -1;
        }
    }
```

Output

```
abhishek-jadhav@abhishek-jadhav-ubuntu:~/Codes/OS Assignments/33232$ ./a.out
Enter the maximum range of Disk: 199
Enter the number of queue requests: 5
Enter the initial head position: 30
Enter the disk positions to be read(queue): 40
20
160
134
11
Disk head moves from position 30 to 40 with Seek 10
Disk head moves from position 40 to 134 with Seek 94
Disk head moves from position 134 to 160 with Seek 26
Disk head moves from position 160 to 11 with Seek 149
Disk head moves from position 11 to 20 with Seek 9
Total Seek Time= 288
Average Seek Time= 57.599998
```