**Experiment-10 [HW]**

**Disk Scheduling using FCFS**

**Code:**

// FCFS Disk Scheduling algorithm

#include <stdio.h>

#include <math.h>

int size = 8;

void FCFS(int arr[], int head)

{

int seek\_count = 0;

int cur\_track, distance;

for (int i = 0; i < size; i++)

{

cur\_track = arr[i];

// calculate absolute distance

distance = fabs(head - cur\_track);

// increase the total count

seek\_count += distance;

// accessed track is now new head

head = cur\_track;

}

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

// Seek sequence would be the same

// as request array sequence

printf("Seek Sequence is\n");

for (int i = 0; i < size; i++)

{

printf("%d\n", arr[i]);

}

}

// Driver code

int main()

{

// request array

int arr[8] = {176, 79, 34, 60, 92, 11, 41, 114};

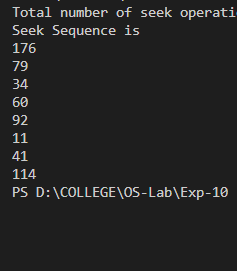
int head = 50;

FCFS(arr, head);

return 0;

}

**Output:**



**Disk Scheduling using Shortest Seek Time First**

**Code:**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int RQ[100], i, n, TotalHeadMoment = 0, initial, count = 0;

printf("Enter the number of Requests\n");

scanf("%d", &n);

printf("Enter the Requests sequence\n");

for (i = 0; i < n; i++)

scanf("%d", &RQ[i]);

printf("Enter initial head position\n");

scanf("%d", &initial);

// logic for sstf disk scheduling

/\* loop will execute until all process is completed\*/

while (count != n)

{

int min = 1000, d, index;

for (i = 0; i < n; i++)

{

d = abs(RQ[i] - initial);

if (min > d)

{

min = d;

index = i;

}

}

TotalHeadMoment = TotalHeadMoment + min;

initial = RQ[index];

// 1000 is for max

// you can use any number

RQ[index] = 1000;

count++;

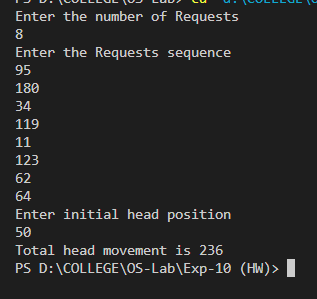
}

printf("Total head movement is %d", TotalHeadMoment);

return 0;

}

**Output:**



**Disk Scheduling using CScan**

**Code:**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int RQ[100], i, j, n, TotalHeadMoment = 0, initial, size, move;

printf("Enter the number of Requests\n");

scanf("%d", &n);

printf("Enter the Requests sequence\n");

for (i = 0; i < n; i++)

scanf("%d", &RQ[i]);

printf("Enter initial head position\n");

scanf("%d", &initial);

printf("Enter total disk size\n");

scanf("%d", &size);

printf("Enter the head movement direction for high 1 and for low 0\n");

scanf("%d", &move);

// logic for C-Scan disk scheduling

/\*logic for sort the request array \*/

for (i = 0; i < n; i++)

{

for (j = 0; j < n - i - 1; j++)

{

if (RQ[j] > RQ[j + 1])

{

int temp;

temp = RQ[j];

RQ[j] = RQ[j + 1];

RQ[j + 1] = temp;

}

}

}

int index;

for (i = 0; i < n; i++)

{

if (initial < RQ[i])

{

index = i;

break;

}

}

// if movement is towards high value

if (move == 1)

{

for (i = index; i < n; i++)

{

TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);

initial = RQ[i];

}

// last movement for max size

TotalHeadMoment = TotalHeadMoment + abs(size - RQ[i - 1] - 1);

/\*movement max to min disk \*/

TotalHeadMoment = TotalHeadMoment + abs(size - 1 - 0);

initial = 0;

for (i = 0; i < index; i++)

{

TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);

initial = RQ[i];

}

}

// if movement is towards low value

else

{

for (i = index - 1; i >= 0; i--)

{

TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);

initial = RQ[i];

}

// last movement for min size

TotalHeadMoment = TotalHeadMoment + abs(RQ[i + 1] - 0);

/\*movement min to max disk \*/

TotalHeadMoment = TotalHeadMoment + abs(size - 1 - 0);

initial = size - 1;

for (i = n - 1; i >= index; i--)

{

TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);

initial = RQ[i];

}

}

printf("Total head movement is %d", TotalHeadMoment);

return 0;

}

**Output:**

