Name : Anupam Kunwar
Reg : 19BCE1369
Week-13

DISK SCHEDULING

Solution:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
int total_min = 9999;
char name[10];
void fcfs(int noq, int qu[noq], int st)
{
int i, s = 0;
for (i = 0; i < noq; i++)
{
s = s + abs(st - qu[i]);
st = qu[i];
}
printf("Total seek time for FCFS is: :%d", s);
if (total_min >= s)
{
total_min = s;
strcpy(name, "FCFS");
}
}
void sstf(int noq, int qu[noq], int st, int visit[8])
{
int min, s = 0, p, i;
while (1)
{
min = 99999;
for (i = 0; i < noq; i++)
if (visit[i] == 0)
{
if (min > abs(st - qu[i]))
{
min = abs(st - qu[i]);
p = i;
}
}
if (min == 99999)
break;
visit[p] = 1;
s = s + min;
st = qu[p];
}
printf("Total seek time for SSTF is: %d", s);
if (total_min >= s)
{
total_min = s;
strcpy(name, "SSTF");
}
}
void scan(int noq, int qu[noq], int st, int ch)
{
int i = 0, j = 0, t = 0;
for (i = 0; i < noq; i++)
for (j = i + 1; j < noq; j++)
if (qu[i] > qu[j])
{
t = qu[i];
qu[i] = qu[j];
qu[j] = t;
}
int s = 0;
for (i = 0; i < noq; i++)
{
if (st < qu[i])
{
for (j = i - 1; j >= 0; j--)
{
s = s + abs(st - qu[j]);
st = qu[j];
}
if (ch == 3)
{
s = s + abs(st - 0);
st = 0;
}
for (j = 1; j < noq; j++)
{
s = s + abs(st - qu[j]);
st = qu[j];
}
break;
}
}
printf("Total seek time for SCAN: %d", s);
if (total_min >= s)
{
total_min = s;
strcpy(name, "SCAN");
}
}
void cscan(int noq, int queue[noq], int head, int max)
```

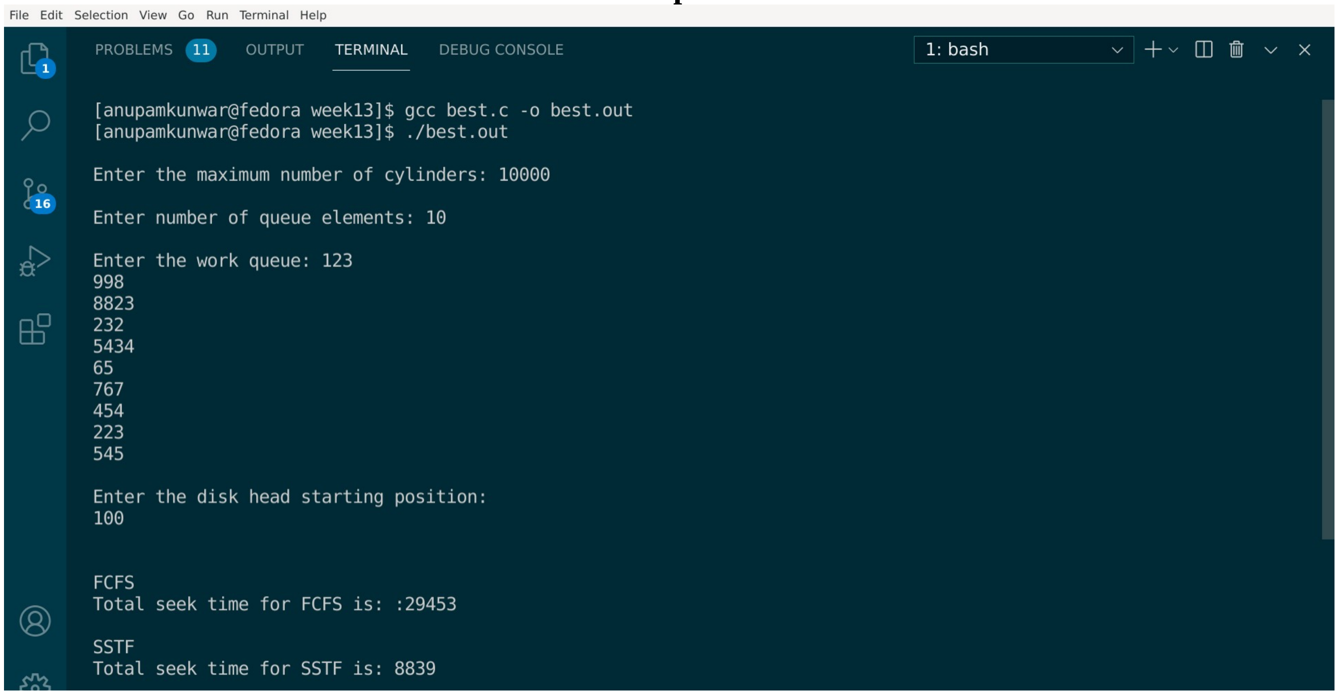
```
{
int
n,
i, j, k, seek = 0, diff, temp, queue1[noq], queue2[noq], temp1 = 0, temp2 = 0;
n = noq;
for (i = 0; i < noq; i++)
{
temp = queue[i];
if (temp >= head)
{
queue1[temp1] = temp;
temp1++;
}
else
{
queue2[temp2] = temp;
temp2++;
}
}
// left sort
for (i = 0; i < temp1 - 1; i++)
{
for (j = i + 1; j < temp1; j++)
{
if (queue1[i] > queue1[j])
{
temp = queue1[i];
queue1[i] = queue1[j];
queue1[j] = temp;
}
}
}
//right sort
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;
}
}
}
int seek_count = 0;
int distance = 0;
int seeksq1[noq];
int pushvar1 = 0;
int cur_track = 0;
for (i = 0; i < temp2; i++)
{
cur_track = queue2[i];
seeksq1[pushvar1] = cur_track;
pushvar1++;
distance = abs(cur_track - head);
seek_count += distance;
head = cur_track;
}
head = 0;
for (i = temp1 - 1; i >= 0; i--)
{
cur_track = queue1[i];
seeksq1[pushvar1] = cur_track;
pushvar1++;
distance = abs(cur_track - head);
seek_count += distance;
head = cur_track;
}
for (j = 0; j < pushvar1 - 1; j++)
{
diff = abs(seeksq1[j + 1] - seeksq1[j]);
seek += diff;
}
printf("Total seek time for CSCAN is %d", seek);
if (total_min >= seek)
{
total_min = seek;
strcpy(name, "CSCAN");
}
}
void look(int noq, int queue[noq], int head, int max, int dir)
{
int
n,
i, j, k, seek = 0, diff, temp, queue1[noq], queue2[noq], temp1 = 0, temp2 = 0;
n = noq;
for (i = 0; i < noq; i++)
{
temp = queue[i];
if (temp >= head)
{
queue1[temp1] = temp;
temp1++;
}
else
{
queue2[temp2] = temp;
```

```
temp2++;
}
}
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;
}
}
}
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;
}
}
}
int run = 2;
int pushvar = 0;
int cur_track = 0;
int distance = 0, seek_count = 0;
int seeksq[noq];
while (run-- > 0)
{
if (dir == 0)
{
for (i = temp1 - 1; i >= 0; i--)
{
cur_track = queue1[i];
seeksq[pushvar] = cur_track;
pushvar++;
distance = abs(cur_track - head);
seek_count += distance;
head = cur_track;
}
dir = 1;
}
else if (dir == 1)
{
```

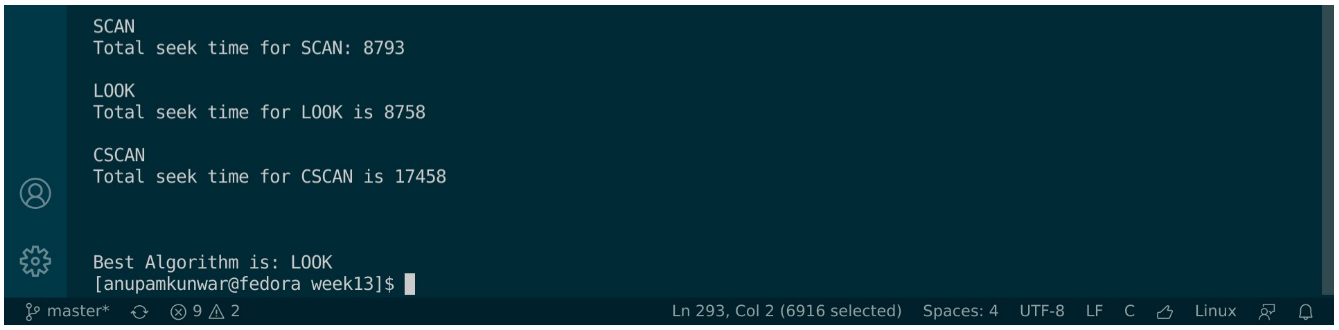
```
for (i = 0; i < temp2; i++)
{
cur_track = queue2[i];
seeksq[pushvar] = cur_track;
pushvar++;
distance = abs(cur_track - head);
seek_count += distance;
head = cur_track;
}
dir = 0;
}
}
for (j = 0; j < pushvar - 1; j++)
{
diff = abs(seeksq[j + 1] - seeksq[j]);
seek += diff;
}
printf("Total seek time for LOOK is %d", seek);
if (total_min >= seek)
{
total_min = seek;
strcpy(name, "LOOK");
}
}
int main()
{
int n, qu[20], st, i, j, t, noq, ch, visit[20];
printf("\nEnter the maximum number of cylinders: ");
scanf("%d", &n);
printf("\nEnter number of queue elements: ");
scanf("%d", &noq);
printf("\nEnter the work queue: ");
for (i = 0; i < noq; i++)
{
scanf("%d", &qu[i]);
visit[i] = 0;
}
printf("\nEnter the disk head starting position: \n");
scanf("%d", &st);
printf("\n\nFCFS \n");
fcfs(noq, qu, st);
printf("\n\nSSTF \n");
sstf(noq, qu, st, visit);
printf("\n\nSCAN \n");
scan(noq, qu, st, ch);
printf("\n\nLOOK \n");
//direction: 1 for right 0 for left
look(noq, qu, st, n, 0);
printf("\n\nCSCAN \n");
```

```
cscan(noq, qu, st, n);
printf("\n \n");
printf("\n\nBest Algorithm is: %s\n", name);
printf("Minimum seek time is: %d", total_min);
return 0;
}
```

Output



```
[anupamkunwar@fedora week13]$ gcc best.c -o best.out
[anupamkunwar@fedora week13]$ ./best.out

Enter the maximum number of cylinders: 10000

Enter number of queue elements: 10

Enter the work queue: 123
998
8823
232
5434
65
767
454
223
545

Enter the disk head starting position:
100


FCFS
Total seek time for FCFS is: :29453

SSTF
Total seek time for SSTF is: 8839
```



```
SCAN
Total seek time for SCAN: 8793

LOOK
Total seek time for LOOK is 8758

CSCAN
Total seek time for CSCAN is 17458


Best Algorithm is: LOOK
[anupamkunwar@fedora week13]$
```