ASSIGNMENT 11

Question:

Implement a program for disk scheduling algorithm and state which is better for number of head movements

1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
5. LOOK
6. C-LOOK

Code:

```c
#include "stdio.h"
#include "stdlib.h"
#include "stdbool.h"

struct request
{
    int request_track_number;
    bool visited;
};

int fcfs()
{
    int i,no_of_requests,initial_head;
    printf("Enter the number of requests: ");
    scanf("%d",&no_of_requests);
    int request[no_of_requests];
    printf("Enter the requests: ");
    for (i = 0; i < no_of_requests; ++i)
    {
        scanf("%d",&request[i]);
    }
    printf("Enter initial position of R/W head: ");
    scanf("%d",&initial_head);
    int seek_time=0;
    printf("%d -> ",initial_head );
    for(i=0;i<no_of_requests;i++)
    {
        if(i == no_of_requests-1)
            printf("%d\n", request[i] );
        else
            printf("%d -> ", request[i] );
```

```
        seek_time += abs(request[i] - initial_head);
        initial_head = request[i];
    }
    printf("Seek Time: %d\n", seek_time);
}

int sstf()
{
    int i,no_of_requests,initial_head,limit,j,choice,previous_head;
    printf("Enter the number of requests: ");
    scanf("%d",&no_of_requests);
    struct request req[no_of_requests];
    printf("Enter the requests: ");
    for (i = 0; i < no_of_requests; ++i)
    {
        scanf("%d",&req[i].request_track_number);
        req[i].visited = false;
    }
    printf("Enter initial position of R/W head: ");
    scanf("%d",&initial_head);

    int seek_time=0;
    printf("%d -> ",initial_head );
    int n = no_of_requests;
    while(n)
    {
        int min = 1e9;
        int min_track_number, position;
        for(i=0;i<no_of_requests;i++)
        {
            if(abs(initial_head - req[i].request_track_number) < min &&
req[i].visited == false)
            {
                min = abs(initial_head - req[i].request_track_number);
                min_track_number = req[i].request_track_number;
                position = i;
            }
        }
        initial_head = req[position].request_track_number;
        req[position].visited = true;
        printf("%d ->",min_track_number);
        seek_time += min;
        n--;
    }

    printf("\nSeek Time: %d\n", seek_time);
}
```

```
int scan()
{
    int i,no_of_requests,initial_head,limit,j,choice,previous_head;
    printf("Enter the number of requests: ");
    scanf("%d",&no_of_requests);
    struct request req[no_of_requests];
    printf("Enter the requests: ");
    for (i = 0; i < no_of_requests; ++i)
    {
        scanf("%d",&req[i].request_track_number);
        req[i].visited = false;
    }
    printf("Enter initial position of R/W head: ");
    scanf("%d",&initial_head);

    printf("Enter the previous position of R/W head: ");
    scanf("%d",&previous_head);

    printf("Enter the cylinder size: ");
    scanf("%d",&limit);

    if(previous_head - initial_head > 0 )
    {
        choice = 2;
    }
    else
        choice = 1;
    //scanf("%d",&choice);
    int seek_time=0;
    printf("%d -> ",initial_head );
    if(choice == 1)
    {
        for(i=initial_head;i<limit;i++)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        printf("%d -> ", limit-1);
```

```
        seek_time += abs(limit-1 - initial_head);
        initial_head = limit-1;
        for(i=initial_head;i>=0;i--)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        seek_time += abs(initial_head - 0);
        printf("0 \n");
    }
    else if(choice == 2)
    {
        for(i=initial_head;i>=0;i--)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        printf("%d -> ", 0);
        seek_time += abs(0 - initial_head);
        initial_head = 0;
        for(i=initial_head;i<limit;i++)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
```

```
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        seek_time += abs(limit-1 - initial_head );
        printf("%d \n", limit-1);

    }
    printf("Seek Time: %d\n", seek_time);
}

int c_scan()
{
    int i,no_of_requests,initial_head,limit,j,choice,previous_head;
    printf("Enter the number of requests: ");
    scanf("%d",&no_of_requests);
    struct request req[no_of_requests];
    printf("Enter the requests: ");
    for (i = 0; i < no_of_requests; ++i)
    {
        scanf("%d",&req[i].request_track_number);
        req[i].visited = false;
    }
    printf("Enter initial position of R/W head: ");
    scanf("%d",&initial_head);

    printf("Enter the previous position of R/W head: ");
    scanf("%d",&previous_head);

    printf("Enter the cylinder size: ");
    scanf("%d",&limit);

    if(previous_head - initial_head > 0 )
    {
        choice = 2;
    }
    else
        choice = 1;
    int seek_time=0;
    printf("%d -> ",initial_head );
    int cp_initial_head = initial_head;
    if(choice == 1)
    {
        for(i=initial_head;i<limit;i++)
        {
```

```
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        printf("%d -> \n", limit-1);
        seek_time += abs(limit-1 - initial_head);
        initial_head = 0;
        for(i=0;i<cp_initial_head;i++)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        printf("\n");
    }
    else if(choice == 2)
    {
        for(i=initial_head;i>=0;i--)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
```

```
            }
        }
        printf("%d -> ", 0 );
        seek_time += abs(initial_head - 0);
        initial_head = limit-1;
        for(i=limit;i>cp_initial_head;i--)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        printf("\n");
    }
    printf("Seek Time: %d\n", seek_time);
}

int look()
{
    int i,no_of_requests,initial_head,limit,j,choice,previous_head;
    printf("Enter the number of requests: ");
    scanf("%d",&no_of_requests);
    struct request req[no_of_requests];
    printf("Enter the requests: ");
    for (i = 0; i < no_of_requests; ++i)
    {
        scanf("%d",&req[i].request_track_number);
        req[i].visited = false;
    }
    printf("Enter initial position of R/W head: ");
    scanf("%d",&initial_head);

    printf("Enter the previous position of R/W head: ");
    scanf("%d",&previous_head);

    printf("Enter the cylinder size: ");
    scanf("%d",&limit);

    if(previous_head - initial_head > 0 )
    {
```

```
        choice = 2;
    }
    else
        choice = 1;
    //scanf("%d",&choice);
    int seek_time=0;
    printf("%d -> ",initial_head );
    if(choice == 1)
    {
        for(i=initial_head;i<limit;i++)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        for(i=initial_head;i>=0;i--)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        printf("\n");
    }
    else if(choice == 2)
    {
        for(i=initial_head;i>=0;i--)
        {
            for(j=0;j<no_of_requests;j++)
            {
```

```
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        for(i=initial_head;i<limit;i++)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        printf("\n");
    }
    printf("Seek Time: %d\n", seek_time);
}

int c_look()
{
    int i,no_of_requests,initial_head,limit,j,choice,previous_head;
    printf("Enter the number of requests: ");
    scanf("%d",&no_of_requests);
    struct request req[no_of_requests];
    printf("Enter the requests: ");
    for (i = 0; i < no_of_requests; ++i)
    {
        scanf("%d",&req[i].request_track_number);
        req[i].visited = false;
    }
    printf("Enter initial position of R/W head: ");
    scanf("%d",&initial_head);

    printf("Enter the previous position of R/W head: ");
    scanf("%d",&previous_head);
```

```
    printf("Enter the cylinder size: ");
    scanf("%d",&limit);

    if(previous_head - initial_head > 0 )
    {
        choice = 2;
    }
    else
        choice = 1;
    //scanf("%d",&choice);
    int seek_time=0;
    printf("%d -> ",initial_head );
    int cp_initial_head = initial_head;
    if(choice == 1)
    {
        for(i=initial_head;i<limit;i++)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        initial_head = 0;
        for(i=0;i<cp_initial_head;i++)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        printf("\n");
```

```
    }
    else if(choice == 2)
    {
        for(i=initial_head;i>=0;i--)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        initial_head = limit-1;
        for(i=limit;i>cp_initial_head;i--)
        {
            for(j=0;j<no_of_requests;j++)
            {
                if(req[j].request_track_number == i && req[j].visited ==
false)
                {
                    printf("%d -> ", req[j].request_track_number);
                    req[j].visited = true;
                    seek_time += abs(req[j].request_track_number -
initial_head);
                    initial_head = req[j].request_track_number;
                }
            }
        }
        printf("\n");
    }
    printf("Seek Time: %d\n", seek_time);
}

void main()
{
    int x;
    printf("1. FCFS\n2. SSTF\n3. SCAN\n4. C-SCAN\n5. LOOK\n6. C-LOOK\nEnter
Your Choice: ");
    scanf("%d", &x);
    switch(x)
    {
        case 1:
```

```
            fcfs();
            break;
        case 2:
            sstf();
            break;
        case 3:
            scan();
            break;
        case 4:
            c_scan();
            break;
        case 5:
            look();
            break;
        case 6:
            c_look();
            break;
        default:
            printf("Enter correct choice!!!");
    }
    printf("Do you want to run again?\n1. Yes\n2. No\nEnter Your Choice: ");
    scanf("%d", &x);
    if (x==1){
        main();
    }
}
```

Output:

```
1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
5. LOOK
6. C-LOOK
Enter Your Choice: 1
Enter the number of requests: 5
Enter the requests: 25 90 33 45 4
Enter initial position of R/W head: 50
50 -> 25 -> 90 -> 33 -> 45 -> 4
Seek Time: 200
Do you want to run again?
1. Yes
2. No
Enter Your Choice: 1
```

```
1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
5. LOOK
6. C-LOOK
Enter Your Choice: 2
Enter the number of requests: 5
Enter the requests: 25 90 33 45 4
Enter initial position of R/W head: 50
50 -> 45 ->33 ->25 ->4 ->90 ->
Seek Time: 132
Do you want to run again?
1. Yes
2. No
Enter Your Choice: 1
```

```
1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
5. LOOK
6. C-LOOK
Enter Your Choice: 3
Enter the number of requests: 5
Enter the requests: 25 90 33 45 4
Enter initial position of R/W head: 50
Enter the previous position of R/W head: 49
Enter the cylinder size: 100
50 -> 90 -> 99 -> 45 -> 33 -> 25 -> 4 -> 0
Seek Time: 148
Do you want to run again?
1. Yes
2. No
Enter Your Choice: 1
```

```
1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
5. LOOK
6. C-LOOK
Enter Your Choice: 4
Enter the number of requests: 5
Enter the requests: 25 90 33 45 4
Enter initial position of R/W head: 50
Enter the previous position of R/W head: 49
Enter the cylinder size: 100
50 -> 90 -> 99 ->
4 -> 25 -> 33 -> 45 ->
Seek Time: 94
Do you want to run again?
1. Yes
2. No
Enter Your Choice: 1
```

```
1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
5. LOOK
6. C-LOOK
Enter Your Choice: 5
Enter the number of requests: 5
Enter the requests: 25 90 33 45 4
Enter initial position of R/W head: 50
Enter the previous position of R/W head: 49
Enter the cylinder size: 100
50 -> 90 -> 45 -> 33 -> 25 -> 4 ->
Seek Time: 126
Do you want to run again?
1. Yes
2. No
Enter Your Choice: 1
```

```
1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
5. LOOK
6. C-LOOK
Enter Your Choice: 6
Enter the number of requests: 5
Enter the requests: 25 90 33 45 4
Enter initial position of R/W head: 50
Enter the previous position of R/W head: 49
Enter the cylinder size: 100
50 -> 90 -> 4 -> 25 -> 33 -> 45 ->
Seek Time: 85
Do you want to run again?
1. Yes
2. No
Enter Your Choice: 2
```