IARE

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING
(An Autonomous Institute affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad - 500 043

LABORATORY WORK BOOK

Name of the Student : RACHERLA SANTHOSH

Roll Number									
2	3	9	5	1	A	1	2	C	3

Class : IT-B Semester : 03

Course Code : ACSD10 Course Name : OS Laboratory

Name of the Course Faculty : Mr. N. Raghava Rao Faculty ID : IARE10924

Exercise Number : 08 Week Number : 08 Date : 08/11/2024

S. No.	Exercise Number	EXERCISE NAME	MARKS AWARDED						
			Aim/ Preparation	Algorithm / Procedure	Performance in the Lab	Source Code	Program Execution	Viva - Voce	Total
						Calculations and Graphs	Results and Error Analysis		
			4	4		4	4	4	20
1	8.1	The Disk Access Dilemma	4		4	4	4	4	20
2	8.2	The SSTF DISK Scheduling Challenge							
3	8.3	FutureTech Corporation							
4	8.4	The C-SCAN DISK SCHEDULING ODYSSEY							
5	8.5	The C-SCAN DISK SCHEDULING Quest.							
6									
7									
8									
9									
10									
11									
12									

Signature of the Student

Signature of the Faculty

8. Disk Scheduling.

8.1 The Disk Access Dilemma :-

AIM :- Write a Program for the disk access dilemma which operates under the First - Come, First Served (FCFS).

PROGRAM :-

```
initial_position = 150
requests = [200, 50, 800, 300, 100]
def fcfs_disk_scheduling (initial_position, requests):
    total_head_movement = 0
    current_position = initial_position
    order_of_tracks = [initial_position]
    for request in requests:
        movement = abs (current_position - request)
        total_head_movement += movement
        current_position = request
        order_of_tracks.append (request)
```

```
    return total_head_movement, order_of_tracks

total_movement, track_order = fcfs_disk_Scheduling
                              (initial_position, requests)
Print ("Total head movement:", total_movement)
Print ("Order of tracks visited:", track_order)
```

OUTPUT :-

Total head movement : 1150

Order of tracks visited : [150, 200, 50, 800, 300, 100]

8.2 The SSTF Disk Scheduling Challenge :-

AIM :- Write a Program on the goal of SSTF is to reduce the total seek time by always selecting the request closest to the current head position. This approach minimizes the movement of the disk head between requests.

ROLL NUMBER :

PROGRAM :-

```
initial_position = 750
requests = [1200, 500, 900, 1500, 300]
def sstf_disk_scheduling (initial_position, requests):
    total_head_movement = 0
    current_position = initial_position
    order_of_tracks = [initial_position]
    remaining_requests = requests.copy()
    while remaining_requests :
        closest_request = min (remaining_requests, key =
        lambda x : abs (current_position - x))
        movement = abs (current_position - closest_request)
        total_head_movement += movement
        current_Position = closest_request
        order_of_tracks.append (closest_request)
        remaining_requests.remove (closest_request)
    return total_head_movement, order_of_tracks
```

```
total_movement, track_order = sstf_disk_scheduling
                              (initial_position, requests)
Print ("Total head movement:", total_movement)
Print ("Order of tracks visited:", track_order)
```

OUTPUT :-

Total head movement : 1950

Order of tracks visited : [750, 900, 1200, 1500, 500, 300]

8.3 Future Tech Corporation :-

AIM :- Write a program on Future Tech Corporation using elevator algorithm.

PROGRAM :-

```
initial_position = 2500
requests = [2800, 1500, 3500, 4000, 1000]
disk_size = 500
def scan_disk_scheduling (initial_position,
    requests, disk_size) :
```

```
requests_above = Sorted ([track for track in requests if
                          track > initial_position])
request_below = Sorted [[track for track in requests if
                         track < initial_position],
                         reverse = true)
total_head_movement = 0
order_of_tracks = []
current_position = initial_position
for track in requests_above :
    movement = abs (current_position - track)
    total_head_movement += movement
    current_position = track
    order_of_tracks.append (track)
return total_head_movement, order_of_tracks
total_movement, track_order = Scan_disk_
    Scheduling (initial_Position, requests, disk_size)
Print (" Total head movement :", total_movement)
Print (" Order of tracks visited :",
```

[Initial_position] + track_order)

OUTPUT :-

Total head movement : 5498

Order of tracks visited : [2500, 2800, 3500, 4000, 4999, 1500, 1000]

8.4 The C-SCAN Disk Scheduling Odyssey :-

AIM :- Write a program for the C-SCAN Disk Scheduling Odyssey.

PROGRAM :-

```
initial_position = 4000
requests = [4200, 1000, 6000, 7500, 2000]
disk_size = 10000
def c_scan_disk_scheduling (initial_
  position, requests, disk_size):
  requests_above = sorted ([track for track in
  requests if track >= initial_position])
```

```
requests_below = sorted([track for track in requests
                         if track < initial_position])
total_head_movement = 0
order_of_tracks = []
current_position = initial_position
for track in requests_above:
    movement = abs(current_position - track)
    total_head_movement += movement
    current_position = track
    order_of_tracks.append(track)
return total_head_movement, order_of_tracks
total_movement, track_order = c_scan_disk_
    scheduling(initial_position, requests, disk_size)
Print("Total head movement:", total_movement)
Print("Order of tracks visited:",
      [initial_position] + track_order)
```

OUTPUT :-

Total head movement : 13999

Order of tracks visited : [4000, 4200, 6000, 7500, 9999, 0, 1000, 2000]

8.5 The C-SCAN Disk Scheduling Quest at Tech fusion Labs :-

AIM :- Write a Program for Tech Fusion Labs on the C-SCAN Disk Scheduling Quest.

PROGRAM :-

```
initial_position = 3500
requests = [3800, 600, 7000, 1500, 2500]
disk_size = 8000
def c_scan_disk_scheduling (initial_
Position, requests, disk_size):
```

```
requests_above = sorted([track for track in
  requests if track >= initial_position])
requests_below = sorted([track for track in
  requests if track < initial_position])
total_head_movement = 0
order_of_tracks = []
current_position = initial_position
for track in requests_above:
  movement = abs(current_position - track)
  total_head_movement += movement
  current_position = track
  order_of_tracks.append(track)
return total_head_movement, order_of_tracks
total_movement, track_order = c_scan_
  disk_scheduling(initial_position, requests,
  disk_size)
```

```
Print ("Total head movement :", total_movement)
Print ("Order of tracks visited :", [initial_position]
        + track_order).
```

OUTPUT :-

```
Total head movement : 14998
Order of tracks visited : [3500, 3800,
6000, 7000, 7999, 0, 600, 1500, 2500]
```