NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

OPERATING SYSTEM

LAB FILE

2021UCA1859

SNEHA GUPTA

Experiment 1

Aim:
Practicing Linux Shell Commands

Objective:
The objective of this experiment is to familiarize yourself with the basic Linux shell commands and their usage.

```
PS C:\Users\lenovo\sneha> pwd

Path
----
C:\Users\lenovo\sneha
```

```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\lenovo> ls


    Directory: C:\Users\lenovo


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         10/5/2022   6:45 PM                .anaconda
d-----         10/5/2022   6:51 PM                .conda
d-----         3/20/2023   7:25 PM                .config
d-----         10/5/2022   6:44 PM                .continuum
d-----         9/29/2022   1:46 PM                .idlerc
d-----         10/5/2022   7:03 PM                .ipython
d-----        12/16/2022   4:09 PM                .redhat
d-----         1/26/2022  12:28 AM                .vscode
d-----         10/5/2022   6:47 PM                anaconda3
d-----         4/14/2023   7:13 PM                anki
d-r---          2/7/2023  10:02 AM                Contacts
d-----         4/14/2023  10:10 PM                Desktop
d-----         9/29/2022   1:59 PM                Documents
d-r---         4/14/2023  10:03 PM                Downloads
d-r---          2/7/2023  10:02 AM                Favorites
d-r---          2/7/2023  10:02 AM                Links
d-r---          2/7/2023  10:02 AM                Music
dar--l         4/13/2023   7:35 PM                OneDrive
d-----         3/27/2023   4:35 PM                Pictures
d-r---          2/7/2023  10:02 AM                Saved Games
d-r---          2/7/2023  10:02 AM                Searches
d-----         4/14/2023  10:11 PM                sneha
d-----         9/29/2022   1:57 PM                source
d-----        11/15/2022  10:41 AM                Tic-Tac-Toe
d-r---          2/7/2023  10:02 AM                Videos
-a----         1/16/2023   3:37 PM            906 .bash_history
-a----         10/5/2022   6:44 PM             25 .condarc
-a----        10/23/2022  10:58 AM            233 .gitconfig
-a----         3/20/2023   7:12 PM              5 .node_repl_history
-a----         1/11/2023   1:07 PM           1774 .viminfo
-a----        12/20/2022  10:59 AM           1128 practical.xml
-a----        12/20/2022  11:02 AM             47 snehapractical.xml
-a----         1/11/2023   1:07 PM             24 test_script


PS C:\Users\lenovo>
```

```
PS C:\Users\lenovo\sneha> mkdir os_prac


    Directory: C:\Users\lenovo\sneha


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         4/14/2023  10:11 PM                os_prac


PS C:\Users\lenovo\sneha> cd os_prac
PS C:\Users\lenovo\sneha\os_prac> mkdir temp


    Directory: C:\Users\lenovo\sneha\os_prac


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         4/14/2023  10:11 PM                temp


PS C:\Users\lenovo\sneha\os_prac> touch first.c
```

```
PS C:\Users\lenovo\sneha\os_prac> cd..
PS C:\Users\lenovo\sneha> rm -r os_prac
PS C:\Users\lenovo\sneha> echo hello world
hello
world
PS C:\Users\lenovo\sneha>
```

Experiment 2

Aim:

Writing C Programs using Fork(), Getpid(), Getppid() and Exec() System Calls

Objective:

The objective of this experiment is to understand and practice the use of the fork(), getpid(), getppid() and exec() system calls in C programming.

main.c

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    pid_t pid, mypid, myppid;
    pid = getpid();
    printf("Before fork: Process id is %d\n",pid);
    pid = fork();
    if (pid < 0) {
    perror("fork() failure\n");
    return 1;
    }
    if (pid==0){
        printf("This is child process\n");
        mypid = getpid();
        myppid = getppid();
        printf("process id is %d and PPID is %d\n", mypid, myppid);
    }
    else { // Parent process
    sleep(2);
    printf("this is parent process\n");
    mypid = getpid();
    myppid = getppid();
    printf("Process id is %d and PPID is %d\n", mypid, myppid);
    printf("New1y created process id or child pid is %d\n"
    , pid);
    }
    return 0;
}
```

Output

```
/tmp/nxKMUnKfgF.o
Before fork: Process id is 351
This is child process
process id is 352 and PPID is 351
this is parent process
Process id is 351 and PPID is 344
New1y created process id or child pid is 352
```

Experiment 3

Aim:

Writing a C Program to Represent a Family of Processes as a Tree

main.c

```c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
    int main(){
    pid_t pid1, pid2, pid3, pid4;
    printf("Parent of all: %d\n",getpid());
    pid1 = fork();
    if(pid1 == 0){
        printf("Child with id: %d and its Parent id: %d \n", getpid(),getppid());
        pid2 = fork();
        if(pid2 == 0){
            printf("Child with id: %d and its Parent id: %d\n", getpid(),getppid());
        }
    }
    if(pid1 > 0){
        pid3 = fork();
        if(pid3 == 0){
            printf("Child with id: %d and its Parent id: %d\n", getpid(),getppid());
            pid4 = fork();
            if(pid4 == 0){
                printf("Child with id: %d and its Parent id: %d\n", getpid(),getppid());
            }
        }
    }
}
```

Output

```
/tmp/nxKMUnKfgF.o
Parent of all: 833
Child with id: 834 and its Parent id: 833
Child with id: 835 and its Parent id: 1
Child with id: 837 and its Parent id: 1
Child with id: 836 and its Parent id: 1
```

Experiment 4

Aim:

Writing a Program to Simulate FCFS CPU Scheduling Algorithm

exp4.cpp > findWaitingTime(int [], int, int [], int [])

```cpp
//by sneha gupta 2021UCA1859
#include<iostream>
using namespace std;
void findWaitingTime(int processes[], int n,int bt[], int wt[])
{
    // waiting time for first process is 0
    wt[0] = 0;
    // calculating waiting time
    for (int i = 1; i < n ; i++ )
    wt[i] = bt[i-1] + wt[i-1] ;
}

void findTurnAroundTime( int processes[], int n, int bt[], int wt[], int tat[])
{
    // calculating turnaround time by adding
    // bt[i] + wt[i]
    for (int i = 0; i < n ; i++)
    tat[i] = bt[i] + wt[i];
}

void findavgTime( int processes[], int n, int bt[])
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    //Function to find waiting time of all processes
    findWaitingTime(processes, n, bt, wt);

    //Function to find turn around time for all processes==
    findTurnAroundTime(processes, n, bt, wt, tat);

    //Display processes along with all details
    cout << "Processes "<< " Burst time "
    << " Waiting time " << " Turn around time\n";
    for (int i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
```

exp4.cpp > findWaitingTime(int [], int, int [], int [])

```cpp
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << " " << i+1 << "\t\t" << bt[i] <<"\t " << wt[i] <<"\t\t "
        << tat[i] <<endl;
    }
    cout << "Average waiting time = "<< (float)total_wt / (float)n;
    cout << "\nAverage turn around time = "<< (float)total_tat / (float)
    n;
    cout << '\n';
}

int main()
{
    int num, temp;

    cout<<"Enter number of Process: ";
    cin>>num;
    int processes[num];
    int burst_time[num];
    cout<<"...Enter the process ID...\n";

    for(int i=0; i<num; i++)
    {
        cout<<"...Process "<<i+1<<"...\n";
        cout<<"Enter Process Id: ";
        cin>>processes[i];
        cout<<"Enter Burst Time: ";
        cin>>burst_time[i];
    }
    int n = sizeof processes / sizeof processes[0];

    findavgTime(processes, n, burst_time);
    return 0;
```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ; if ($?) { g++ exp4.cpp -o exp4 } ; if ($?) { .\exp4 }
Enter number of Process: 5
...Enter the process ID...
...Process 1...
Enter Process Id: 3
Enter Burst Time: 4
...Process 2...
Enter Process Id: 1
Enter Burst Time: 2
...Process 3...
Enter Process Id: 4
Enter Burst Time: 5
...Process 4...
Enter Process Id: 5
Enter Burst Time: 6
...Process 5...
Enter Process Id: 6
Enter Burst Time: 7
Processes  Burst time  Waiting time  Turn around time
 1             4        0              4
 2             2        4              6
 3             5        6              11
 4             6        11             17
 5             7        17             24
Average waiting time = 7.6
Average turn around time = 12.4
PS C:\sem 4\OS\os prac>
```

Experiment 5

Aim:

Writing a Program to Simulate SJF Scheduling Algorithm

exp5.cpp > completionTime(int, int [][6])

```cpp
#include<iostream>
using namespace std;
int mat[10][6];
void swap(int *a, int *b)
{
int temp = *a;
*a = *b;
*b = temp;
}
void arrangeArrival(int num, int mat[][6])
{
for(int i=0; i<num; i++)
{
for(int j=0; j<num-i-1; j++)
{
if(mat[j][1] > mat[j+1][1])
{
for(int k=0; k<5; k++)
{
swap(mat[j][k], mat[j+1][k]);
}
}
}
}
}
void completionTime(int num, int mat[][6])
{
int temp, val;
mat[0][3] = mat[0][1] + mat[0][2];
```

exp5.cpp > completionTime(int, int [][6])

```cpp
{
int temp, val;
mat[0][3] = mat[0][1] + mat[0][2];
mat[0][5] = mat[0][3] - mat[0][1];
mat[0][4] = mat[0][5] - mat[0][2];
for(int i=1; i<num; i++)
{
temp = mat[i-1][3];
int low = mat[i][2];
for(int j=i; j<num; j++)
{
if(temp >= mat[j][1] && low >= mat[j][2])
{
low = mat[j][2];
val = j;
}
}
mat[val][3] = temp + mat[val][2];
mat[val][5] = mat[val][3] - mat[val][1];
mat[val][4] = mat[val][5] - mat[val][2];
for(int k=0; k<6; k++)
{
swap(mat[val][k], mat[i][k]);
}
}
}
int main()
{
int num, temp;
cout<<"Enter number of Process: ";
```

exp5.cpp > completionTime(int, int [][6])

```
cout<<"Enter number of Process: ";
cin>>num;
cout<<"...Enter the process ID...\n";
for(int i=0; i<num; i++)
{
cout<<"...Process "<<i+1<<"...\n";
cout<<"Enter Process Id: ";
cin>>mat[i][0];
cout<<"Enter Arrival Time: ";
cin>>mat[i][1];
cout<<"Enter Burst Time: ";
cin>>mat[i][2];
}
cout<<"Before Arrange...\n";
cout<<"Process ID\tArrival Time\tBurst Time\n";
for(int i=0; i<num; i++)
{
cout<<mat[i][0]<<"\t\t"<<mat[i][1]<<"\t\t"<<mat[i]
[2]<<"\n";
}
arrangeArrival(num, mat);
completionTime(num, mat);
cout<<"Final Result...\n";
cout<<"Process ID\tArrival Time\tBurst Time\tWaiting
Time\tTurnaround Time\n";
for(int i=0; i<num; i++)
{
cout<<mat[i][0]<<"\t\t"<<mat[i][1]<<"\t\t"<<mat[i]
[2]<<"\t\t"<<mat[i][4]<<"\t\t"<<mat[i][5]<<"\n";
}
```

exp5.cpp > completionTime(int, int [][6])

```cpp
for(int i=0; i<num; i++)
{
cout<<mat[i][0]<<"\t\t"<<mat[i][1]<<"\t\t"<<mat[i]
[2]<<"\t\t"<<mat[i][4]<<"\t\t"<<mat[i][5]<<"\n";
}
int avgw = 0;
int avgt = 0;
for(int i=0; i<num; i++)
{
avgw += mat[i][4];
avgt += mat[i][5];
}
cout << "Average waiting time = "
<< float(avgw) / float(num);
cout << "\nAverage turnaround time = "
<< float(avgt) / float(num) << endl;
}
```

```
4\OS\os prac\" ; if ($?) { g++ exp5.cpp -o exp5 } ; if ($?) { .\exp5 }
Enter number of Process: 5
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Arrival Time: 2
Enter Burst Time: 2
...Process 2...
...Process 3...
Enter Process Id: 4
Enter Arrival Time: 5
Enter Burst Time: 6
...Process 4...
Enter Process Id: 7
Enter Arrival Time: 8
Enter Burst Time: 9
...Process 5...
Enter Process Id: 5
Enter Arrival Time: 6
Enter Burst Time: 7
Before Arrange...
Process ID      Arrival Time    Burst Time
1               2               2
2               3               4
4               5               6
7               8               9
5               6               7
Final Result...
Process ID      Arrival Time    Burst Time      Waiting Time        Turnaround Time
1               2               2               0  22               3               4               1  54              5               6
            3  95               6               7               8  15
7               8               9               13 22
Average waiting time = 5
Average turnaround time = 10.6
PS C:\sem 4\OS\os prac>
```

Experiment 6

Aim:

Writing a Program to Simulate preemptive Priority CPU scheduling algorithm

exp6.cpp > main()

```cpp
#include<bits/stdc++.h>
using namespace std;
struct Process
{
int pid;
int bt;
int priority;
};
bool comparison(Process a, Process b)
{
return (a.priority > b.priority);
}
void findWaitingTime(Process proc[], int n,
int wt[])
{
wt[0] = 0;
for (int i = 1; i < n ; i++ )
wt[i] = proc[i-1].bt + wt[i-1] ;
}
void findTurnAroundTime( Process proc[], int n,
int wt[], int tat[])
{
for (int i = 0; i < n ; i++)
tat[i] = proc[i].bt + wt[i];
}
void findavgTime(Process proc[], int n)
{
int wt[n], tat[n], total_wt = 0, total_tat = 0;
findWaitingTime(proc, n, wt);
findTurnAroundTime(proc, n, wt, tat);
```

exp6.cpp > main()

```
cout << "\nProcesses "<< " Burst time "
<< " Waiting time " << " Turn around time\n";
for (int i=0; i<n; i++)
{
total_wt = total_wt + wt[i];
total_tat = total_tat + tat[i];
cout << " " << proc[i].pid << "\t\t"
<< proc[i].bt << "\t " << wt[i]
<< "\t\t " << tat[i] <<endl;
}
cout << "\nAverage waiting time = "
<< (float)total_wt / (float)n;
cout << "\nAverage turn around time = "
<< (float)total_tat / (float)n;
}
void priorityScheduling(Process proc[], int n)
{
sort(proc, proc + n, comparison);
cout<< "Order in which processes gets executed \n";
for (int i = 0 ; i < n; i++)
cout << proc[i].pid <<" " ;
findavgTime(proc, n);
}
int main()
{
    int num, temp;

    cout<<"Enter number of Process: ";
    cin>>num;
    int processes[num];
```

exp6.cpp > main()

```cpp
    int processes[num];
int burst_time[num];
    int priority[num];
    cout<<"...Enter the process ID...\n";
    for(int i=0; i<num; i++)
    {
        cout<<"...Process "<<i+1<<"...\n";
        cout<<"Enter Process Id: ";
        cin>>processes[i];
        cout<<"Enter Burst Time: ";
        cin>>burst_time[i];
        cout<<"Enter Priority: ";
        cin>>priority[i];
    }
    Process proc [num];
    for(int i=0; i<num; i++)
    {
        proc[i] = {processes[i], burst_time[i], priority
        [i]};
    }
    int n = sizeof(proc) / sizeof(proc[0]);
priorityScheduling(proc, n);
return 0;
}
```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ; if ($?) { g++ exp6.cpp -o exp6 } ; if ($?) { .\exp6 }
Enter number of Process: 5
...Process 1...
Enter Process Id: 1 2
Enter Burst Time: Enter Priority: 2
...Process 2...
Enter Process Id: 3
Enter Burst Time: 4
Enter Priority: 5
...Process 3...
Enter Process Id: 5
Enter Burst Time: 6
Enter Priority: 7
...Process 4...
Enter Process Id: 8
Enter Burst Time: 9
Enter Priority: 2
...Process 5...
Enter Process Id: 4
Enter Burst Time: 5
Enter Priority: 2
Order in which processes gets executed
5 3 1 8 4
Processes  Burst time  Waiting time  Turn around time
 5             6        0              6
 3             4        6              10
 1             2        10             12
 8             9        12             21
 4             5        21             26

Average waiting time = 9.8
Average turn around time = 15
PS C:\sem 4\OS\os prac>
```

Experiment 7

Aim:

Writing a Program to Simulate Round Robin Scheduling Algorithm Objective:

exp7.cpp > findWaitingTime(int [], int, int [], int [], int)

```cpp
#include<iostream>
using namespace std;
void findWaitingTime(int processes[], int n,
int bt[], int wt[], int quantum)
{
int rem_bt[n]; // array of remaining times
for (int i = 0 ; i < n ; i++)
rem_bt[i] = bt[i];
int t = 0; // current time
while (1) // traversing
{
bool done = true;
for (int i = 0 ; i < n; i++)
        {
if (rem_bt[i] > 0)
{
done = false; // There is a pending process
if (rem_bt[i] > quantum)
{
// how much time a process has been processed
t += quantum;
// Decrease the burst_time of current process
rem_bt[i] -= quantum;
}
else
{
// how much time a process has been processed
t = t + rem_bt[i];
wt[i] = t - bt[i];
rem_bt[i] = 0;
```

exp4.cpp exp5.cpp exp6.cpp exp7.cpp

exp7.cpp > findWaitingTime(int [], int, int [], int [], int)

```cpp
rem_bt[i] = 0;
}
}
}
if (done == true)
break;
}
}
void findTurnAroundTime(int processes[], int n,
int bt[], int wt[], int tat[])
{
// calculating turnaround time by adding
// bt[i] + wt[i]
for (int i = 0; i < n ; i++)
tat[i] = bt[i] + wt[i];
}
void findavgTime(int processes[], int n, int bt[],
int quantum)
{
int wt[n], tat[n], total_wt = 0, total_tat = 0;
findWaitingTime(processes, n, bt, wt, quantum);
findTurnAroundTime(processes, n, bt, wt, tat);
cout << "Processes "<< " Burst time "
<< " Waiting time " << " Turn around time\n";
for (int i=0; i<n; i++)
{
total_wt = total_wt + wt[i];
total_tat = total_tat + tat[i];
cout << " " << i+1 << "\t\t" << bt[i] <<"\t "
<< wt[i] <<"\t\t " << tat[i] <<endl;
```

exp7.cpp > findWaitingTime(int [], int, int [], int [], int)

```
<< wt[i] <<"\t\t " << tat[i] <<endl;
}
cout << "Average waiting time = "
<< (float)total_wt / (float)n;
cout << "\nAverage turn around time = "
<< (float)total_tat / (float)n;
}
int main()
{
int num, temp;

    cout<<"Enter number of Process: ";
    cin>>num;
    int processes[num];
int burst_time[num];
    cout<<"...Enter the process ID...\n";
    for(int i=0; i<num; i++)
    {
        cout<<"...Process "<<i+1<<"...\n";
        cout<<"Enter Process Id: ";
        cin>>processes[i];
        cout<<"Enter Burst Time: ";
        cin>>burst_time[i];
    }
int n = sizeof processes / sizeof processes[0];
int quantum;
    cout<<"Enter Quantum: ";
    cin>>quantum;
findavgTime(processes, n, burst_time, quantum);
return 0;
```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp7.cpp -o exp7 } ; if ($?) { .\exp7
}
Enter number of Process: 4
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Burst Time: 2
...Process 2...
Enter Process Id: 3
Enter Burst Time: 4
...Process 3...
Enter Process Id: 5
Enter Burst Time: 56
...Process 4...
Enter Process Id: 6
Enter Burst Time: 7
Enter Quantum: 7
Processes  Burst time  Waiting time  Turn around tim
e
 1              2        0              2
 2              4        2              6
 3              56       13             69
 4              7        13             20
Average waiting time = 7
Average turn around time = 24.25
PS C:\sem 4\OS\os prac>
```

Experiment 8

Aim:

Writing a Program to Simulate Multilevel Feedback Queue Scheduling Algorithm

exp8.cpp > main()

```
//code by sneha gupta 2021UCA1859
#include<stdio.h>
struct process
{
    char name;
    int AT,BT,WT,TAT,RT,CT;
}Q1[10],Q2[10],Q3[10];/*Three queues*/
int n;
void sortByArrival()
{
struct process temp;
int i,j;
for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
            {
                if(Q1[i].AT>Q1[j].AT)
                    {
                        temp=Q1[i];
                        Q1[i]=Q1[j];
                        Q1[j]=temp;
                    }
            }
    }
}
int main()
{
    int i,j,k=0,r=0,time=0,tq1=5,tq2=8,flag=0;
    char c;
    printf("Enter no of processes:");
```

exp8.cpp > main()

```
        char c;
        printf("Enter no of processes:");
        scanf("%d",&n);
        for(i=0,c='A';i<n;i++,c++)
        {
            Q1[i].name=c;
            printf("\nEnter the arrival time and burst time of process %c: ",Q1[i].name);
            scanf("%d%d",&Q1[i].AT,&Q1[i].BT);
            Q1[i].RT=Q1[i].BT;/*save burst time in remaining time for
each process*/
        }
sortByArrival();
time=Q1[0].AT;
printf("Process in first queue following RR with qt=5");
printf("\nProcess\t\tRT\t\tWT\t\tTAT\t\t");
for(i=0;i<n;i++)
{
  if(Q1[i].RT<=tq1)
  {
        time+=Q1[i].RT;/*from arrival time of first process to
completion of this process*/
        Q1[i].RT=0;
        Q1[i].WT=time-Q1[i].AT-Q1[i].BT;/*amount of time process
has been waiting in the first queue*/
        Q1[i].TAT=time-Q1[i].AT;/*amount of time to execute
```

exp8.cpp > main()

```cpp
            Q2[i].RT=0;
            Q2[i].WT=time-tq1-Q2[i].BT;/*amount of time process has
been waiting in the ready queue*/
            Q2[i].TAT=time-Q2[i].AT;/*amount of time to execute the
process*/
            printf("\n%c\t\t%d\t\t%d\t\t%d",Q2[i].name,Q2[i].BT,Q2[i].WT,Q2[i].TAT);
        }
        else/*process moves to queue 3 with FCFS*/
        {
          Q3[r].AT=time;
          time+=tq2;
          Q2[i].RT-=tq2;
          Q3[r].BT=Q2[i].RT;
          Q3[r].RT=Q3[r].BT;
          Q3[r].name=Q2[i].name;
          r=r+1;
          flag=2;
        }
      }
    {if(flag==2)
    printf("\nProcess in third queue following FCFS ");
    }
    for(i=0;i<r;i++)
    {
        if(i==0)
                Q3[i].CT=Q3[i].BT+time-tq1-tq2;
            else
```

exp8.cpp > main()

```
            Q1[i].TAT=time-Q1[i].AT;/*amount of time to execute the
process*/
            printf("\n%c\t\t%d\t\t%d\t\t%d",Q1[i].name,Q1[i].BT,Q1[i].WT,Q1[i].TAT);
        }
        else/*process moves to queue 2 with qt=8*/
        {
            Q2[k].WT=time;
            time+=tq1;
            Q1[i].RT-=tq1;
            Q2[k].BT=Q1[i].RT;
            Q2[k].RT=Q2[k].BT;
            Q2[k].name=Q1[i].name;
            k=k+1;
            flag=1;
        }
    }
    if(flag==1)
    {printf("\nProcess in second queue following RR with qt=8");
        printf("\nProcess\t\tRT\t\tWT\t\tTAT\t\t");
    }for(i=0;i<k;i++)
        {
         if(Q2[i].RT<=tq2)
          {
            time+=Q2[i].RT;/*from arrival time of first process +BT of
this process*/
            Q2[i].RT=0;
```

```
            Q3[i].CT=Q3[i].BT+time-tq1-tq2;
        else
            Q3[i].CT=Q3[i-1].CT+Q3[i].BT;
}
for(i=0;i<r;i++)
    {
        Q3[i].TAT=Q3[i].CT;
        Q3[i].WT=Q3[i].TAT-Q3[i].BT;
        printf("\n%c\t\t%d\t\t%d\t\t%d\t\t",Q3[i].name,Q3
        [i].BT,Q3[i
].WT,Q3[i].TAT);
    }
}
/*
Q1 is round robin with time quantum 5 (RR5),
Q2 is round robin with time quantum 8 (RR8), and
Q3 follows first come first serve (FCFS)*/
```

```
TERMINAL                                                    Code

Average turn around timecd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp8.cpp -o exp8 } ; if ($?) { .\exp8
}
Enter no of processes:5

Enter the arrival time and burst time of process A:
2
3

Enter the arrival time and burst time of process B:
4
5

Enter the arrival time and burst time of process C:
5
2

Enter the arrival time and burst time of process D:
1
2

Enter the arrival time and burst time of process E:
1
3
Process in first queue following RR with qt=5
Process         RT              WT              TAT
D               2               0               2
E               3               2               5
A               3               4               7
B               5               5               10
C               2               9               11
PS C:\sem 4\OS\os prac>
```

Experiment 9

Aim:

Writing a Program to Simulate Deadlock Avoidance

exp9.cpp > ...

```cpp
//code by sneha gupta 2021UCA1859

#include <iostream>
using namespace std;
int main()
{
int n, m, i, j, k;
    cout<<"Enter number of Processes: "; // 5
    cin>>n;
    cout<<"Enter number of Resources: "; // 3
    cin>>m;
int alloc[n][m];
    /*{ { 0, 1, 0 }, // P0
{ 2, 0, 0 }, // P1
{ 3, 0, 2 }, // P2
{ 2, 1, 1 }, // P3
{ 0, 0, 2 } }; // P4 */
    i=0;
    j=0;
    while (i < n) {
        cout << "Enter the allocation for " << i + 1 << "
        Process: ";
        while (j < m) {
            cin >> alloc[i][j];
            j++;
        }
        i++;
        j=0;
    }
int max[n][m];
```

exp6.cpp exp7.cpp exp8.cpp exp9.cpp

exp9.cpp > ...

```cpp
    }
int max[n][m];
    /*{ { 7, 5, 3 }, // P0
        { 3, 2, 2 }, // P1
        { 9, 0, 2 }, // P2
        { 2, 2, 2 }, // P3
        { 4, 3, 3 } }; // P4 */
    i=0;
    j=0;
    while (i < n) {
        cout << "Enter the MAX allocation for " << i + 1
        << " Process: ";
        while (j < m) {
            cin >> max[i][j];
            j++;
        }
    i++;
    j=0;
    }

int avail[m];
    //{ 3, 3, 2 }
    cout<<"Enter Available Units of Resources: ";
    for (i=0;i<m;i++){
        cin>>avail[i];
        }
    // BANKERS ALGORITHM
int f[n], ans[n], ind = 0;
for (k = 0; k < n; k++) {
f[k] = 0;
```

exp9.cpp > main()

```
f[k] = 0;
}
int need[n][m];
for (i = 0; i < n; i++) {
for (j = 0; j < m; j++)
need[i][j] = max[i][j] - alloc[i][j];
}
int y = 0;
for (k = 0; k < 5; k++) {
for (i = 0; i < n; i++) {
if (f[i] == 0) {
int flag = 0;
for (j = 0; j < m; j++) {
if (need[i][j] > avail[j]){
flag = 1;
break;
}
}
if (flag == 0) {
ans[ind++] = i;
for (y = 0; y < m; y++)
avail[y] += alloc[i][y];
f[i] = 1;
}
}
}
}
cout << "Following is the required Sequence" << endl;
for (i = 0; i < n - 1; i++)
cout << " P" << ans[i] << " ->";
```

```
avail[y] += alloc[i][y];
f[i] = 1;
}
}
}
}
cout << "Following is the required Sequence" << endl;
for (i = 0; i < n - 1; i++)
cout << " P" << ans[i] << " ->";
cout << " P" << ans[n - 1] <<endl;
return (0);
}
```

```
Average turn around timecd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp8.cpp -o exp8 } ; if ($?) { .\exp8
}
}
Enter number of Processes: 5
Enter number of Resources: 1
Enter the allocation for 1 Process: 2
Enter the allocation for 2 Process: 3
Enter the allocation for 3 Process: 4
Enter the allocation for 4 Process: 4
Enter the allocation for 5 Process: 5
Enter the MAX allocation for 1 Process: 6
Enter the MAX allocation for 2 Process: 7
Enter the MAX allocation for 3 Process: 5
Enter the MAX allocation for 4 Process: 4
Enter the MAX allocation for 5 Process: 6
Enter Available Units of Resources: 3
Following is the required Sequence
 P2 -> P3 -> P4 -> P0 -> P1
```

Experiment 10

Aim:

Writing a Program to Simulate Deadlock Detection

exp10.cpp > main()

```c
//code by sneha gupta 2021UCA1859
#include<stdio.h>
static int mark[20];
int i,j,np,nr;
int main()
{
int alloc[10][10],request[10][10],avail[10],r[10],w[10];
printf("\nEnter the no of process: ");
scanf("%d",&np);
printf("\nEnter the no of resources: ");
scanf("%d",&nr);
for(i=0;i<nr;i++)
{
printf("\nTotal Amount of the Resource R%d: ",i+1);
scanf("%d",&r[i]);
}
printf("\nEnter the request matrix:");
for(i=0;i<np;i++)
for(j=0;j<nr;j++)
scanf("%d",&request[i][j]);
printf("\nEnter the allocation matrix:");
for(i=0;i<np;i++)
for(j=0;j<nr;j++)
scanf("%d",&alloc[i][j]);
/*Available Resource calculation*/
for(j=0;j<nr;j++)
{
avail[j]=r[j];
for(i=0;i<np;i++)
{
```

exp10.cpp > main()

```cpp
for(i=0;i<np;i++)
{
avail[j]-=alloc[i][j];
}
}
//marking processes with zero allocation
for(i=0;i<np;i++)
{
int count=0;
for(j=0;j<nr;j++)
    {
        if(alloc[i][j]==0)
          count++;
        else
          break;
     }
if(count==nr)
mark[i]=1;
}
// initialize W with avail
for(j=0;j<nr;j++)
     w[j]=avail[j];
//mark processes with request less than or equal to W
for(i=0;i<np;i++)
{
int canbeprocessed=0;
if(mark[i]!=1)
{
    for(j=0;j<nr;j++)
     {
```

exp10.cpp > main()

```
        for(j=0;j<nr;j++)
         {
           if(request[i][j]<=w[j])
             canbeprocessed=1;
           else
               {
               canbeprocessed=0;
               break;
                }
          }
    if(canbeprocessed)
    {
    mark[i]=1;
    for(j=0;j<nr;j++)
    w[j]+=alloc[i][j];
    }
    }
    }
    //checking for unmarked processes
    int deadlock=0;
    for(i=0;i<np;i++)
    if(mark[i]!=1)
    deadlock=1;
    if(deadlock)
    printf("\n Deadlock detected\n");
    else
    printf("\n No Deadlock possible\n");
    }

```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp10.cpp -o exp10 } ; if ($?) { .\exp
10 }

Enter the no of process: 5

Enter the no of resources: 1

Total Amount of the Resource R1: 2

Enter the request matrix:3
4
5
6
7

Enter the allocation matrix:4
3
2
6
3

 Deadlock detected
PS C:\sem 4\OS\os prac> 5
```

Experiment 11

Aim: **Write a program to simulate best-fit contiguous memory allocation.**

exp8.cpp | exp9.cpp 4 | exp10.cpp | exp11.cpp

exp11.cpp > bestFit(int [], int, int [], int)

```cpp
//code by sneha gupta 2021UCA1859

#include<bits/stdc++.h>
using namespace std;
void bestFit(int blockSize[], int m, int processSize[], int n)
{

int allocation[n];
memset(allocation, -1, sizeof(allocation));
// pick each process and find suitable blocks
// according to its size ad assign to it
for (int i=0; i<n; i++)
{
// Find the best fit block for current process
int bestIdx = -1;
for (int j=0; j<m; j++)
{
if (blockSize[j] >= processSize[i])
{
if (bestIdx == -1)
bestIdx = j;
else if (blockSize[bestIdx] > blockSize[j])
bestIdx = j;
}
}
// If we could find a block for current process
if (bestIdx != -1)
{
// allocate block j to p[i] process
```

exp11.cpp > bestFit(int [], int, int [], int)

```cpp
{
// allocate block j to p[i] process
allocation[i] = bestIdx;
// Reduce available memory in this block.
blockSize[bestIdx] -= processSize[i];
}
}
cout << "\nProcess No.\tProcess Size\tBlock no.\n";
for (int i = 0; i < n; i++)
{
cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";
if (allocation[i] != -1)
cout << allocation[i] + 1;
else
cout << "Not Allocated";
cout << endl;
}
}
int main()
{
    int m, n;
    cout<<"Enter the number of blocks: ";
    // 5
    cin>>m;
    cout<<"Enter the number of processes: ";
    // 4
    cin>>n;
int blockSize[m];
int processSize[n];
    cout<<"Enter the block sizes: ";
```

```
int blockSize[m];
int processSize[n];
    cout<<"Enter the block sizes: ";
    // 300 200 600 100 500
    for (int temp=0;temp<m;temp++){
        cin>>blockSize[temp];
    }
    cout<<"Enter the process sizes: ";
    // 212 417 112 426
    for (int temp=0;temp<n;temp++){
        cin>>processSize[temp];
    }
bestFit(blockSize, m, processSize, n);
return 0 ;
}
```

```
 Deadlock detected
PS C:\sem 4\OS\os prac> 5
5                       cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp11.cpp -o exp11 } ; if ($?) { .\exp
11 }
Enter the number of blocks: 5
Enter the number of processes:
4
Enter the block sizes: 2
5
6
3
6
Enter the process sizes: 3
6
3
6

Process No.     Process Size    Block no.
 1              3               4
 2              6               3
 3              3               2
 4              6               5
```

Experiment 12

Aim: Write a program to simulate FIFO page replacement algorithm

exp12.cpp > main()

```cpp

//code by sneha gupta 2021UCA1859

#include<bits/stdc++.h>
using namespace std;
int pageFaults(int pages[], int n, int capacity)
{
    /*
    Returns page faults
    args:
    pages -- array of all pages that will be called
    n -- number of pages
    capacity -- number of pages that can be held in memory
    */
    // set to hold current set of pages
unordered_set<int> s;
// Queue to implement FIFO
queue<int> indexes;
// init_ variables
int page_faults = 0;
for (int i=0; i<n; i++)
{
// Check if the set can hold more pages
if (s.size() < capacity)
{
// Insert it into set if not present; represents page fault
if (s.find(pages[i])==s.end())
{
// Insert the current page into the set
s.insert(pages[i]);
```

exp12.cpp > main()

```
// Insert the current page into the set
s.insert(pages[i]);
// increment page fault
page_faults++;
// Push the current page into the queue
indexes.push(pages[i]);
}
}
// If the set is full then need to perform FIFO
// remove the first page and insert the current page
else
{
// Check if current page is present
if (s.find(pages[i]) == s.end())
{
// Store the first page
int val = indexes.front();
// Pop the first page from the queue
indexes.pop();
// Remove the indexes page from the set
s.erase(val);
// insert the current page in the set
s.insert(pages[i]);
// push the current page
indexes.push(pages[i]);
// Increment page faults
page_faults++;
}
}
}
```

exp12.cpp > main()

```cpp
}
}
}
return page_faults;
}
int main()
{
    int n;
    cout<<"Enter the number of pages in the sequence: ";
    // 13
    cin>>n;
    int pages[n];
    cout<<"Enter the sequence of pages that will be called: ";
    // 7 0 1 2 0 3 0 4 2 3 0 3 2
    for (int i=0; i<n; i++){
        cin>>pages[i];
    }
int capacity;
    // 4
    cout<<"Enter the capacity of pages that can be held in memory: ";
    cin>>capacity;
cout << "The number of page faults encountered: " <<
pageFaults(pages, n, capacity);
    cout<<'\n';
return 0;
}
```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp12.cpp -o exp12 } ; if ($?) { .\exp
12 }
Enter the number of pages in the sequence: 5
Enter the sequence of pages that will be called: 4
6
7
4
5
Enter the capacity of pages that can be held in memo
ry: 3
The number of page faults encountered: 4
```

Experiment 13

Aim: Write a program to simulate LRU page replacement algorithm

exp13.cpp > ...

```cpp
//code by sneha gupta 2021UCA1859

#include<bits/stdc++.h>
using namespace std;
int pageFaults(int pages[], int n, int capacity)
{
    /*
    returns -- Number of page faults

Args:
    pages -- list of referenced pages
n -- number of pages
capacity -- number of pages that can be simultaneously in
memory
    */
unordered_set<int> s;
unordered_map<int, int> indexes;
int page_faults = 0;
for (int i=0; i<n; i++)
{
// Check if the set can hold more pages
if (s.size() < capacity)
{
if (s.find(pages[i])==s.end())
{
s.insert(pages[i]);
page_faults++;
}
indexes[pages[i]] = i;
}
```

exp13.cpp > ...

```cpp
}
indexes[pages[i]] = i;
}
else
{
if (s.find(pages[i]) == s.end())
{
int lru = INT_MAX, val;
for (auto it=s.begin(); it!=s.end(); it++)
{
if (indexes[*it] < lru)
{
lru = indexes[*it];
val = *it;
}
}
// Remove the indexes page
s.erase(val);
// insert the current page
s.insert(pages[i]);
// Increment page faults
page_faults++;
}
// Update the current page index
indexes[pages[i]] = i;
}
}
return page_faults;
}
int main()
```

```
}
}
return page_faults;
}
int main()
{
int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};
int n = sizeof(pages)/sizeof(pages[0]);
int capacity = 4;
cout << pageFaults(pages, n, capacity);
cout<<endl;
return 0;
}
```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp13.cpp -o exp13 } ; if ($?) { .\exp
13 }
6
PS C:\sem 4\OS\os prac>
```

Experiment 14

Aim: Write a program to simulate Second Chance page replacement algorithm.

exp14.cpp > ...

```cpp
//code by sneha gupta 2021UCA1859

#include<iostream>
#include<cstring>
#include<sstream>
using namespace std;
static bool findAndUpdate(int x,int arr[],bool
second_chance[],int frames)
{
int i;
for(i = 0; i < frames; i++)
{
if(arr[i] == x)
{
second_chance[i] = true;
return true;
}
}
return false;
}
static int replaceAndUpdate(int x,int arr[],bool
second_chance[],int frames,int pointer)
{
while(true)
{
if(!second_chance[pointer])
{
arr[pointer] = x;
return (pointer + 1) % frames;
}
```

exp11.cpp exp13.cpp exp10.cpp exp14.cpp

exp14.cpp > ...

```cpp
}
second_chance[pointer] = false;
pointer = (pointer + 1) % frames;
}
}
static void printHitsAndFaults(string reference_string,int
frames)
{
int pointer, i, l=0, x, pf;
pointer = 0;
pf = 0;
int arr[frames];
memset(arr, -1, sizeof(arr));
bool second_chance[frames];
string str[100];
string word = "";
for (auto x : reference_string)
{
if (x == ' ')
{
str[l]=word;
word = "";
l++;
}
else
{
word = word + x;
}
}
str[l] = word;
```

exp11.cpp exp13.cpp exp10.cpp exp14.cpp

exp14.cpp > ...

```cpp
str[l] = word;
l++;
// l=the length of array
for(i = 0; i < l; i++)
{
x = stoi(str[i]);
// Finds if there exists a need to replace
// any page at all
if(!findAndUpdate(x,arr,second_chance,frames))
{
// Selects and updates a victim page
pointer = replaceAndUpdate(x,arr,
second_chance,frames,pointer);
// Update page faults
pf++;
}
}
cout << "Total page faults were " << pf << "\n";
}
int main()
{
string reference_string = "";
int frames = 0;
// Test 1:
reference_string = "0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4";
frames = 3;
// Output is 9
printHitsAndFaults(reference_string,frames);
// Test 2:
```

```
printHitsAndFaults(reference_string,frames);
// Test 2:
reference_string = "2 5 10 1 2 2 6 9 1 2 10 2 6 1 2 1 6 9 5 1";
frames = 4;
// Output is 11
printHitsAndFaults(reference_string,frames);
return 0;
}
```

```
Average turn around time = 13.2
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
14 }
Total page faults were 9
Total page faults were 11
PS C:\sem 4\OS\os prac>
```

Experiment 15

Aim: Write a program to simulate Enhanced Second Chance page replacement algorithm.

exp15.cpp > main()

```cpp
#include <bits/stdc++.h>
#define float double
#define sz 100005
#define all(a) a.begin(), a.end()
#define mod 1000000007
using namespace std;
#define vi vector<int>
#define vvi vector<vector<int>>
#define debug cout << "here" << endl;
#define rep(i, n) for (int i = 0; i < n; ++i)
#define pb push_back
#define ff first
#define ss second
#define pi pair<int, int>
using namespace std;
struct req
{
    int id;
    bool modified;
    bool used;
};
void simulate(vector<req> &request, int numRequest, int
numFrames)
{
    int id = 0;
    int pageFaultCount = 0;
    map<int, bool> inMemory;
    int filled = 0;
    vector<req> frames(numFrames, {-1, 0, 0});
    for (auto node : request)
```

exp15.cpp > main()

```cpp
    int pageFaultCount = 0;
    map<int, bool> inMemory;
    int filled = 0;
    vector<req> frames(numFrames, {-1, 0, 0});
    for (auto node : request)
    {
        cout << node.id << " ----> ";
        filled++;
        if (inMemory[node.id])
        {
            rep(i, numFrames)
            {
                if (frames[i].id == node.id)
                {
                    frames[i].used = 1;
                    inMemory[node.id] = true;
                    break;
                }
            }
        }
        else if (filled <= numFrames)
        {
            cout << "(PAGE FAULT) ";
            pageFaultCount++;
            rep(i, numFrames)
            {
                if (frames[i].id == -1)
                {
                    frames[i] = {node.id, node.modified,
                    1};
                    inMemory[node.id] = true;
```

exp15.cpp > main()

```cpp
                    frames[i] = {node.id, node.modified, 1};
                    inMemory[node.id] = true;
                    break;
                }
            }
        }
        else
        {
            cout << "(PAGE FAULT) ";
            pageFaultCount++;
            inMemory[node.id] = true;
            bool done = false;
            rep(p, numFrames)
            {
                if (frames[id].used == false && frames[id].modified == false)
                {
                    inMemory[frames[id].id] = false;
                    frames[id] = {node.id, node.modified, 1};
                    done = true;
                    id = (id + 1) % numFrames;
                    break;
                }
                else
                {
                    id = (id + 1) % numFrames;
                }
```

exp15.cpp > main()

```cpp
                    id = (id + 1) % numFrames;
                }
            }
            if (!done)
            {
                rep(p, 2 * numFrames)
                {
                    if (frames[id].used == false && frames[id].modified == 1)
                    {
                        inMemory[frames[id].id] = false;
                        frames[id] = {node.id, node.modified, 1};
                        done = true;
                        id = (id + 1) % numFrames;
                        break;
                    }
                    else
                    {
                        frames[id].used = false;
                        id = (id + 1) % numFrames;
                    }
                }
            }
        }
        cout << "[ ";
        for (auto val : frames)
        {
            if (val.id == -1)
```

exp15.cpp > main()

```cpp
            cout << "[ ";
            for (auto val : frames)
            {
                if (val.id == -1)
                    cout << " - ";
                else
                    cout << val.id << " ";
            }
            cout << "]";
            cout << endl;
        }
        cout << "Total number of page faults are " <<
    pageFaultCount << endl;
    }
    int32_t main()
    {
        int numFrames;
        cout << "Enter the number of frames" << endl;
        cin >> numFrames;
        cout << "Enter the number memory request " << endl;
        int numRequest;
        cin >> numRequest;
        cout << "Enter the request" << endl;
        vector<req> request(numRequest, {0, 0, 0});
        rep(i, numRequest) cin >> request[i].id;
        cout << "Enter which frames have been modified " <<
        endl;
        rep(i, numRequest) cin >> request[i].modified;
        simulate(request, numRequest, numFrames);
```

```
Total page faults were 11
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp15.cpp -o exp15 } ; if ($?) { .\exp
15 }
Enter the number of frames
3
Enter the number memory request
4
Enter the request
5
6
2
4
Enter which frames have been modified
2
5 ----> (PAGE FAULT) [ 5  -  - ]
6 ----> (PAGE FAULT) [ 5 6  - ]
2 ----> (PAGE FAULT) [ 5 6 2 ]
4 ----> (PAGE FAULT) [ 4 6 2 ]
Total number of page faults are 4
PS C:\sem 4\OS\os prac> 4
```

Experiment 15

Aim: Write a program to simulate LFU page replacement algorithm.

exp16.cpp > ...

```cpp

//code by sneha gupta 2021UCA1859

#include<bits/stdc++.h>
using namespace std;
int main(){
    int n,m,i,j,k;
    cout<<"Enter number of frames\n";
    cin>>n;
    cout<<"Enter number of processes\n";
    cin>>m;
    vector<int> p(m);
    cout<<"Enter processes\n";
    for(i=0;i<m;i++){
        cin>>p[i];
    }
    vector<vector<int>> a(n,vector<int>(m,-1));
    map <int, int> mp,lfmp;
    for(i=0;i<m;i++){
        vector<int> op;
        vector<pair<int,int>> c,lf;
        for(auto q: mp){
            c.push_back({q.second,q.first});
        }
        for(auto q:lfmp){
            lf.push_back({q.second,q.first});
        }
        sort(lf.begin(),lf.end());
        bool dontCall=true;
        if(lf.size()>2){
```

exp16.cpp > ...

```cpp
        bool dontCall=true;
        if(lf.size()>2){
            if(lf[0].first!=lf[1].first){
                dontCall=false;
            }
        }
        sort(c.begin(),c.end());
        bool hasrun=false;
        for(j=0;j<n;j++){
            if(a[j][i]==p[i]){
                mp[p[i]]++;
                lfmp[p[i]]++;
                hasrun=true;
                break;
            }
            if(a[j][i]==-1){
                for(k=i;k<m;k++)
                    a[j][k]=p[i];
                mp[p[i]]++;
                lfmp[p[i]]++;
                hasrun=true;
                break;
            }
        }
        if(j==n||hasrun==false){
            for(j=0;j<n;j++){
                if(dontCall==true){
                    int q;
                    if(lf[lf.size()-1].second==c[c.size()
                    -1].second&&lf[lf.
size()-1].first>1){
```

Run Testcases ⊗ 4 ⚠ 0 Ln 4, Col 1 Spaces: 4 UTF-8 CRLF {} C++ Go Live

exp16.cpp > ...

```
                    int q;
                    if(lf[lf.size()-1].second==c[c.size()-1].second&&lf[lf.size()-1].first>1){
                        if(a[j][i]==c[c.size()-2].second){
                        mp.erase(a[j][i]);
                        lfmp.erase(a[j][i]);
                        for(k=i;k<m;k++)
                            a[j][k]=p[i];
                        mp[p[i]]++;
                        lfmp[p[i]]++;
                        break;
                        }

                    }
                else{
                    if(a[j][i]==c[c.size()-1].second){
                        mp.erase(a[j][i]);
                        lfmp.erase(a[j][i]);
                        for(k=i;k<m;k++)
                            a[j][k]=p[i];
                        mp[p[i]]++;
                        lfmp[p[i]]++;
                        break;
                    }
                    }
                }
                else if(dontCall==false){
                    if(a[j][i]==lf[0].second){
                        mp.erase(a[j][i]);
                        lfmp.erase(a[j][i]);
```

exp16.cpp > ...

```cpp
                    lfmp.erase(a[j][i]);
                    for(k=i;k<m;k++)
                        a[j][k]=p[i];
                    mp[p[i]]++;
                    lfmp[p[i]]++;
                    break;
                }
            }
        }
    }
    for(auto q:mp){
        if(q.first!=p[i]){
            mp[q.first]++;
        }
    }
}
int hit=0;
vector<int> hitv(m);
for(i=1;i<m;i++){
    for(j=0;j<n;j++){
        if(p[i]==a[j][i-1]){
            hit++;
            hitv[i]=1;
            break;
        }
    }
}
cout<<"Process ";
for(i=0;i<m;i++){
    cout<<p[i]<<" ";
```

Run Testcases ⊗ 4 ⚠ 0 Ln 4, Col 1 Spaces: 4 UTF-8 CRLF {} C++ Go Live

exp16.cpp > ...

```cpp
    cout<<"Process ";
    for(i=0;i<m;i++){
        cout<<p[i]<<" ";
    }
    cout<<'\n';
    for(i=0;i<n;i++){
        cout<<"Frame "<<i<<" ";
        for(j=0;j<m;j++){
            if(a[i][j]==-1)
                cout<<"E ";
                else
            cout<<a[i][j]<<" ";
        }
        cout<<'\n';
    }
    cout<<"HIT ";
    for(i=0;i<hitv.size();i++){
        if(hitv[i]==0)
            cout<<" ";
        else
        cout<<hitv[i]<<" ";
    }
    cout<<"\n";
    cout<<"Hit "<<hit<<'\n'<<"Page Fault "<<m-hit<<'\n';
    return 0;
}
```

```
}
Enter number of frames
4
Enter number of processes
6
Enter processes
2
5
2
5
3
5
Process 2 5 2 5 3 5
Frame 0 2 2 2 2 2 2
Frame 1 E 5 5 5 5 5
Frame 2 E E E E 3 3
Frame 3 E E E E E E
HIT   1 1  1
Hit 3
Page Fault 3
PS C:\sem 4\OS\os prac> 3
```

Experiment 17

Aim: Write a program to simulate FCFS disk scheduling algorithm.

exp17.cpp > main()

```cpp
#include <bits/stdc++.h>
using namespace std;
void FCFS(int arr[], int head, int size)
{
int seek_count = 0;
int distance, cur_track;
for (int i = 0; i < size; i++) {
cur_track = arr[i];
// calculate absolute distance
distance = abs(cur_track - head);
// increase the total count
seek_count += distance;
// accessed track is now new head
head = cur_track;
}
cout << "Total number of seek operations = "
<< seek_count << endl;
// Seek sequence would be the same
// as request array sequence
cout << "Seek Sequence is" << endl;
for (int i = 0; i < size; i++) {
cout << arr[i] << endl;
}
}
int main()
{
    int size;
    cout<<"Enter the number of seek addresses: ";
    // 8
    cin>>size;
```

```cpp
    // 8
    cin>>size;
    int arr[size];
    cout<<"Enter the addresses: ";
    for (int i=0;i<size;i++){
        cin>>arr[i];
    }
    // 176 79 34 60 92 11 41 114
int head;
    cout<<"Enter the initial head position: ";
    cin>>head;
    // 50
FCFS(arr, head, size);
return 0;
}
```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp17.cpp -o exp17 } ; if ($?) { .\exp
17 }
Enter the number of seek addresses: 8
Enter the addresses: 4
5
6
7
4
4
6
67
Enter the initial head position: 54
Total number of seek operations = 119
Seek Sequence is
4
5
6
7
4
4
6
67
```

Experiment 18

Aim: Write a program to simulate SSTF disk scheduling algorithm.

exp18.cpp > main()

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,k,n,m,sum=0,x,y,h;
    cout<<"Enter the size of disk\n";
    cin>>m;
    cout<<"Enter number of requests\n";
    cin>>n;
    cout<<"Enter the requests\n";
    //creating two arrays, array a will store the input
    //I/O requests and array b will store the output
    vector <int> a(n),b;
    //creating a map to store the count of each element
    //in the array a.
    map <int,int> mp;
    for(i=0;i<n;i++){
        cin>>a[i];
        mp[a[i]]++;
    }
    for(i=0;i<n;i++){
        if(a[i]>m){
            cout<<"Error, Unknown position "<<a[i]<<"\n";
            return 0;
        }
    }
    cout<<"Enter the head position\n";
    cin>>h;
    int temp=h;
    int ele;
    b.push_back(h);
```

exp18.cpp > main()

```cpp
    b.push_back(h);
    int count=0;
    while(count<n){
        //initially taking diff to be very large.
        int diff=999999;
        //traversing in map to find the least difference
        for(auto q:mp){
            if(abs(q.first-temp)<diff){
                ele=q.first;
                diff=abs(q.first-temp);
            }
        }
        //deleting the element that has the least
        //difference from the map
        mp[ele]--;
        if(mp[ele]==0){
            mp.erase(ele);
        }
        //adding that element to our output array.
        b.push_back(ele);
        temp=ele;
        count++;
    }
    //printing the output array
    cout<<b[0];
    temp=b[0];
    for(i=1;i<b.size();i++){
        cout<<" -> "<<b[i];
        sum+=abs(b[i]-temp);
        temp=b[i];
```

Run Testcases ⊗ 4 ⚠ 0 Ln 65, Col 2 Spaces: 4 UTF-8 CRLF {} C++ Go Live

```
        cout<<" -> "<<b[i];
        sum+=abs(b[i]-temp);
        temp=b[i];
    }
    cout<<'\n';
    cout<<"Total head movements = "<< sum<<'\n';
    cout<<"Average head movement = "<<(float)sum/n<<'\n';
    return 0;
}
```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp18.cpp -o exp18 } ; if ($?) { .\exp
18 }
Enter the size of disk
199
Enter number of requests
5
Enter the requests
12
45
3
5
3
Enter the head position
5
5 -> 5 -> 3 -> 3 -> 12 -> 45
Total head movements = 44
Average head movement = 8.8
```

Experiment 19

Aim: Write a program to simulate C-SCAN disk scheduling algorithm.

exp19.cpp > ...

```cpp
//code by sneha gupta 2021UCA1859

#include <bits/stdc++.h>
using namespace std;
void CSCAN(int arr[], int head, int size, int disk_size)
{
int seek_count = 0;
int distance, cur_track;
vector<int> left, right;
vector<int> seek_sequence;
left.push_back(0);
right.push_back(disk_size - 1);
for (int i = 0; i < size; i++) {
if (arr[i] < head)
left.push_back(arr[i]);
if (arr[i] > head)
right.push_back(arr[i]);
}
std::sort(left.begin(), left.end());
std::sort(right.begin(), right.end());
for (int i = 0; i < right.size(); i++) {
cur_track = right[i];
seek_sequence.push_back(cur_track);
distance = abs(cur_track - head);
seek_count += distance;
head = cur_track;
}
head = 0;
seek_count += (disk_size - 1);
for (int i = 0; i < left.size(); i++) {
```

exp19.cpp > CSCAN(int [], int, int, int)

```cpp
seek_count += (disk_size - 1);
for (int i = 0; i < left.size(); i++) {
cur_track = left[i];
seek_sequence.push_back(cur_track);
distance = abs(cur_track - head);
seek_count += distance;
head = cur_track;
}
cout << "Total number of seek operations = "
<< seek_count << endl;
cout << "Seek Sequence is" << endl;
for (int i = 0; i < seek_sequence.size(); i++) {
cout << seek_sequence[i] << endl;
}
}
int main()
{
    int size;
    cout<<"Enter the number of requests: ";
    cin>>size;
    // 8
    int disk_size;
    cout<<"Enter the disk size: ";
    cin>> disk_size;
    cout<<"Enter the request list: ";
    int arr[size];
    for (int i=0; i<size; i++){
        cin>>arr[i];
    }
// 176 79 34 60 92 11 41 114
```

```
    }
// 176 79 34 60 92 11 41 114
int head;
    cout<<"Enter head position: ";
    cin>>head;
    //50
CSCAN(arr, head, size, disk_size);
return 0;
}
```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp19.cpp -o exp19 } ; if ($?) { .\exp
19 }
Enter the number of requests: 5
Enter the disk size: 55
Enter the request list: 3
5
3
5
3
Enter head position: 5
Total number of seek operations = 106
Seek Sequence is
54
0
3
3
3
```

Experiment 20

Aim: Write a program to simulate LOOK disk scheduling algorithm.

exp20.cpp > ...

```cpp
//code by sneha gupta 2021UCA1859

#include <bits/stdc++.h>
using namespace std;
void LOOK(int arr[], int head, string direction, int size)
{
int seek_count = 0;
int distance, cur_track;
vector<int> left, right;
vector<int> seek_sequence;
for (int i = 0; i < size; i++) {
if (arr[i] < head)
left.push_back(arr[i]);
if (arr[i] > head)
right.push_back(arr[i]);
}
std::sort(left.begin(), left.end());
std::sort(right.begin(), right.end());
int run = 2;
while (run--) {
if (direction == "left") {
for (int i = left.size() - 1; i >= 0; i--) {
cur_track = left[i];
seek_sequence.push_back(cur_track);
distance = abs(cur_track - head);
seek_count += distance;
head = cur_track;
}
direction = "right";
```

exp20.cpp > ...

```cpp
}
direction = "right";
}
else if (direction == "right") {
for (int i = 0; i < right.size(); i++) {
cur_track = right[i];
seek_sequence.push_back(cur_track);
distance = abs(cur_track - head);
seek_count += distance;
head = cur_track;
}
direction = "left";
}
}
cout << "Total number of seek operations = "
<< seek_count << endl;
cout << "Seek Sequence is" << endl;
for (int i = 0; i < seek_sequence.size(); i++) {
cout << seek_sequence[i] << endl;
}
}
int main()
{
    int size;
    cout<<"Enter the number of requests: ";
    cin>>size;
    // 8
    cout<<"Enter the request list: ";
    int arr[size];
    for (int i=0; i<size; i++){
```

Run Testcases ⊗ 4 ⚠ 0 Ln 2, Col 1 Spaces: 4 UTF-8 CRLF {} C++ Go Live

```
    int arr[size];
    for (int i=0; i<size; i++){
        cin>>arr[i];
    }
// 176 79 34 60 92 11 41 114
int head;
    cout<<"Enter head position: ";
    cin>>head;
    //50
string direction = "right";
LOOK(arr, head, direction, size);
return 0;
}
```

```
PS C:\sem 4\OS\os prac> cd "c:\sem 4\OS\os prac\" ;
if ($?) { g++ exp20.cpp -o exp20 } ; if ($?) { .\exp
20 }
Enter the number of requests: 4
Enter the request list: 2
3
4
5
Enter head position: 5
Total number of seek operations = 3
Seek Sequence is
4
3
2
```