INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN

Operating Systems Practical File

BIT-202

Submitted By:
Name: Anushka Kumari
Enrollment No.: 05201012020
Batch: CSE1, B1
B.Tech, 4th Semester

Submitted To:
Dr. Arunima Jaiswal
Assistant Professor
CSE Department
IGDTUW

Index

S.No.	Experiment Name	Date
1.	Execute various LINUX commands: (i) clear, cal, who, date, pwd (ii) cd, mkdir, rmdir, wc, cat	24-01-2022
2.	Execute various LINUX commands: (i) ps, alias, ls, grep (ii) cmp, comm, diff (iii) chmod, chown, chgrp	31-01-2022
3.	To write shell script to find average of three numbers.	14-02-2022
4.	Write a program in C to implement First Come First Serve algorithm.	14-02-2022
5.	Write a program in C to implement Shortest Job First Algorithm.	21-02-2022
6.	Write a program in C to implement Shortest Run Time Next.	21-02-2022
7.	Write a program in C to implement Priority Based Preemption algorithm.	28-02-2022
8.	Write a program in C to implement Priority Based Non - Preemption algorithm.	28-02-2022
9.	Write a program in C to implement Round Robin algorithm.	07-03-2022
10.	Write a program in C to implement Banker's algorithm.	07-03-2022
11.	Write a program in C to implement Optimal Page Replacement algorithm.	11-04-2022
12.	Write a program in C to implement FIFO algorithm.	11-04-2022
13.	Write a program in C to implement LRU algorithm.	18-04-2022
14.	Write a program in C to implement SCAN Disk Scheduling algorithm.	18-04-2022
15.	Write a program in C to implement Shortest Seek Time First Disk Scheduling algorithm.	25-04-2022
16.	Write a program in C to implement C-SCAN Disk Scheduling algorithm.	25-04-2022

Experiment 1

1. Execute various LINUX commands:

1.1

(i) cal

```
abhishek@abhishek-Inspiron-3542:~$ cal
   February 2022
Su Mo Tu We Th Fr Sa
        1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28
```

(ii) who

```
abhishek@abhishek-Inspiron-3542:~$ who
abhishek :0           2022-02-20 22:04 (:0)
```

(iii) date

```
abhishek@abhishek-Inspiron-3542:~$ date
Sun Feb 20 18:50:31 IST 2022
```

(iv) pwd

```
abhishek@abhishek-Inspiron-3542:~$ pwd
/home/abhishek
```

(v) clear

```
File  Edit  View  Search  Terminal  Help
abhishek@abhishek-Inspiron-3542:~/Desktop$
```

1.2

(i) cd, mkdir , rmdir

```
abhishek@abhishek-Inspiron-3542:~$ cd Desktop
abhishek@abhishek-Inspiron-3542:~/Desktop$ cd ..
abhishek@abhishek-Inspiron-3542:~$ cd ..
abhishek@abhishek-Inspiron-3542:/home$ cd ~
abhishek@abhishek-Inspiron-3542:~$ cd Desktop
abhishek@abhishek-Inspiron-3542:~/Desktop$ mkdir testFolder
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls
 AutomaticEssayGrading-master     gephi-0.9.2-linux
 CommunityDetectionCodes-master   LSH-community-detection-master
'csv datasets'                    testFolder
abhishek@abhishek-Inspiron-3542:~/Desktop$ rmdir testFolder
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls
 AutomaticEssayGrading-master     gephi-0.9.2-linux
 CommunityDetectionCodes-master   LSH-community-detection-master
'csv datasets'
```

(ii) wc

```
abhishek@abhishek-Inspiron-3542:~/Desktop$ wc abc.txt
 1  2 30 abc.txt
abhishek@abhishek-Inspiron-3542:~/Desktop$
```

(iii) cat

```
abhishek@abhishek-Inspiron-3542:~/Desktop$ ^C
abhishek@abhishek-Inspiron-3542:~/Desktop$ cat
test
test


cgh
cgh
i am inside $HOME
i am inside $HOME
^C
```

Experiment 2

2.Execute various linux commands;

2.1

(i)ps

```
abhishek@abhishek-Inspiron-3542:~/Desktop$ ps
  PID TTY          TIME CMD
 8233 pts/0    00:00:00 bash
 9073 pts/0    00:00:00 ps
```

(ii)alias

```
abhishek@abhishek-Inspiron-3542:~$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo
 error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'
')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
abhishek@abhishek-Inspiron-3542:~$
```

(iii)ls

```
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls
 AutomaticEssayGrading-master     gephi-0.9.2-linux              testf
 CommunityDetectionCodes-master   gtest
'csv datasets'                    LSH-community-detection-master
```

(iv)grep

```
abhishek@abhishek-Inspiron-3542:~/Desktop$ echo $HOME
/home/abhishek
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls | grep test
abhishek@abhishek-Inspiron-3542:~/Desktop$ mkdir testf
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls | grep test
testf
abhishek@abhishek-Inspiron-3542:~/Desktop$ mkdir gtest
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls | grep test
gtest
testf
```

2.2 cmp, comm, diff

```
abhishek@abhishek-Inspiron-3542:~$ cd Desktop
abhishek@abhishek-Inspiron-3542:~/Desktop$ nano abc.txt
abhishek@abhishek-Inspiron-3542:~/Desktop$ abc.txt
abc.txt: command not found
abhishek@abhishek-Inspiron-3542:~/Desktop$ nano abc.txt
abhishek@abhishek-Inspiron-3542:~/Desktop$ nano ct.txt
abhishek@abhishek-Inspiron-3542:~/Desktop$ cmp abc.txt ct.txt
abc.txt ct.txt differ: byte 1, line 1
abhishek@abhishek-Inspiron-3542:~/Desktop$ comm abc.txt ct.txt
fasgxwhdsjlkfcieujc nerkuhc,e
        hjjkkdfjkkaksaagvvereeenbcxxc
abhishek@abhishek-Inspiron-3542:~/Desktop$ diff abc.txt ct.txt
1c1
< fasgxwhdsjlkfcieujc nerkuhc,e
---
> hjjkkdfjkkaksaagvvereeenbcxxc
abhishek@abhishek-Inspiron-3542:~/Desktop$
```

2.3

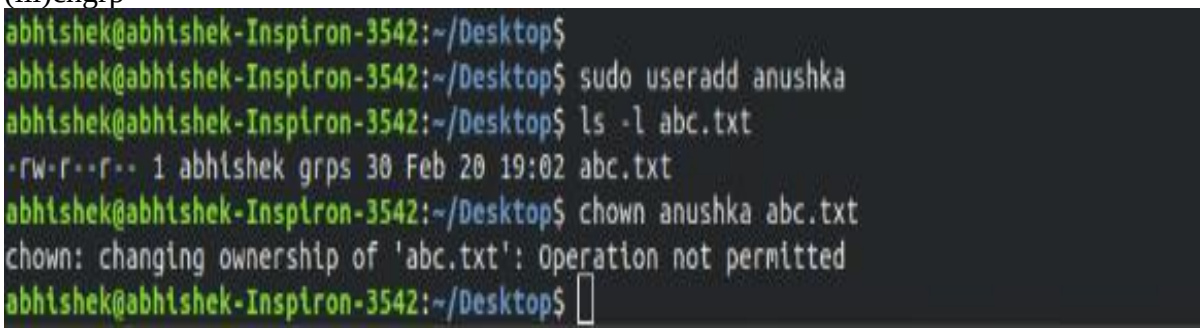
(i) chmod

```
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls
 abc.txt                          gephi-0.9.2-linux
 AutomaticEssayGrading-master     gtest
 CommunityDetectionCodes-master   LSH-community-detection-master
'csv datasets'                    testf
 ct.txt
abhishek@abhishek-Inspiron-3542:~/Desktop$ chmod +x ct.txt
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls
 abc.txt                          gephi-0.9.2-linux
 AutomaticEssayGrading-master     gtest
 CommunityDetectionCodes-master   LSH-community-detection-master
'csv datasets'                    testf
 ct.txt
abhishek@abhishek-Inspiron-3542:~/Desktop$
```

(ii)chown

```
abhishek@abhishek-Inspiron-3542:~/Desktop$ sudo addgroup grp
Adding group `grp' (GID 1002) ...
Done.
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls -l
total 36
-rw-r--r--  1 abhishek grps       30 Feb 20 19:02  abc.txt
drwxrwxr-x  5 abhishek abhishek 4096 Oct 31  2016  AutomaticEssayGrading-master
drwxrwxrwx 11 abhishek abhishek 4096 Mar 19  2019  CommunityDetectionCodes-master
drwxrwxrwx  3 abhishek abhishek 4096 May 24  2019 'csv datasets'
-rwxr-xr-x  1 abhishek abhishek   30 Feb 20 19:04  ct.txt
drwxr-xr-x  3 abhishek abhishek 4096 May 24  2019  gephi-0.9.2-linux
drwxr-xr-x  2 abhishek abhishek 4096 Feb 20 18:56  gtest
drwxrwxr-x  8 abhishek abhishek 4096 Nov 21  2017  LSH-community-detection-master
drwxr-xr-x  2 abhishek abhishek 4096 Feb 20 18:55  testf
abhishek@abhishek-Inspiron-3542:~/Desktop$ sudo chgrp grps abc.txt
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls -l
total 36
-rw-r--r--  1 abhishek grps       30 Feb 20 19:02  abc.txt
drwxrwxr-x  5 abhishek abhishek 4096 Oct 31  2016  AutomaticEssayGrading-master
drwxrwxrwx 11 abhishek abhishek 4096 Mar 19  2019  CommunityDetectionCodes-master
drwxrwxrwx  3 abhishek abhishek 4096 May 24  2019 'csv datasets'
-rwxr-xr-x  1 abhishek abhishek   30 Feb 20 19:04  ct.txt
drwxr-xr-x  3 abhishek abhishek 4096 May 24  2019  gephi-0.9.2-linux
drwxr-xr-x  2 abhishek abhishek 4096 Feb 20 18:56  gtest
drwxrwxr-x  8 abhishek abhishek 4096 Nov 21  2017  LSH-community-detection-master
drwxr-xr-x  2 abhishek abhishek 4096 Feb 20 18:55  testf
abhishek@abhishek-Inspiron-3542:~/Desktop$
```

(iii)chgrp


abhishek@abhishek-Inspiron-3542:~/Desktop$
abhishek@abhishek-Inspiron-3542:~/Desktop$ sudo useradd anushka
abhishek@abhishek-Inspiron-3542:~/Desktop$ ls -l abc.txt
-rw-r--r-- 1 abhishek grps 30 Feb 20 19:02 abc.txt
abhishek@abhishek-Inspiron-3542:~/Desktop$ chown anushka abc.txt
chown: changing ownership of 'abc.txt': Operation not permitted
abhishek@abhishek-Inspiron-3542:~/Desktop$

Experiment 3

Q. To write shell script to find average of three numbers.

```
#shell program to find average of three numbers

clear
echo "Enter 1st number"
read a
echo "Enter 2nd number"
read b
echo "enter 3rd number"
read c
sum='expr $a + $b + $c'
avg='expr $sum/3'
echo "Sum=$sum"
echo "Average=$avg"
```

```
Enter first number:
256
Enter second number:
32
Enter third number:
100
Sum = 388
Average = 129
```

Experiment 4

Q. Write a program in C to implement First Come First Serve algorithm.

Code:

```
#include <stdio.h>
using namespace std;

int waitingtime(int proc[], int n,
   int burst_time[], int wait_time[]) {
   wait_time[0] = 0;
   for (int i = 1; i < n ; i++ )
   wait_time[i] = burst_time[i-1] + wait_time[i-1] ;
   return 0;
}

int turnaroundtime( int proc[], int n,
int burst_time[], int wait_time[], int tat[]) {
    int i;
    for ( i = 0; i < n ; i++)
    tat[i] = burst_time[i] + wait_time[i];
    return 0;
}

int avgtime( int proc[], int n, int burst_time[]) {
    int wait_time[n], tat[n], total_wt = 0, total_tat = 0;
    int i;
    waitingtime(proc, n, burst_time, wait_time);
    turnaroundtime(proc, n, burst_time, wait_time, tat);
    printf("Processes  Burst   Waiting Turn around \n");

   for ( i=0; i<n; i++) {
      total_wt = total_wt + wait_time[i];
      total_tat = total_tat + tat[i];
      printf(" %d\t  %d\t\t %d \t%d\n", i+1, burst_time[i], wait_time[i], tat[i]);
   }
   printf("Average waiting time = %f\n", (float)total_wt / (float)n);
   printf("Average turn around time = %f\n", (float)total_tat / (float)n);
   return 0;
}
int main() {
   printf ("ANUSHKA _ 052 \n" );
   int proc[] = { 1, 2, 3};
   int n = sizeof proc / sizeof proc[0];
   int burst_time[] = {9, 7, 14};
   avgtime(proc, n, burst_time);
    return 0;
}
```

Output:

```
ANUSHKA _ 052
Processes  Burst   Waiting Turn around
 1        9              0      9
 2        7              9      16
 3        14             16     30
Average waiting time = 8.333333
Average turn around time = 18.333333

Process returned 0 (0x0)   execution time : 0.055 s
Press any key to continue.
```

Experiment 5

Q. Write a program in C to implement Shortest Job First Algorithm.

Code:

```
#include<stdio.h>
using namespace std;

int main()
{
   int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
   float avg_wt,avg_tat;
   printf (“ANUSHKA _ 052 \n”) ;
   printf("Enter number of process:");
   scanf("%d",&n);

   printf("\nEnter Burst Time:\n");
   for(i=0;i<n;i++)
   {
      printf("p%d:",i+1);
      scanf("%d",&bt[i]);
      p[i]=i+1;
   }

   for(i=0;i<n;i++)
   {
      pos=i;
      for(j=i+1;j<n;j++)
      {
         if(bt[j]<bt[pos])
            pos=j;
      }

      temp=bt[i];
      bt[i]=bt[pos];
      bt[pos]=temp;

      temp=p[i];
      p[i]=p[pos];
      p[pos]=temp;
   }

   wt[0]=0;
   for(i=1;i<n;i++)
   {
      wt[i]=0;
      for(j=0;j<i;j++)
```

```
            wt[i]+=bt[j];

        total+=wt[i];
    }

    avg_wt=(float)total/n;
    total=0;

    printf("\nProcess\t    Burst Time \t Waiting Time\t  Turnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=(float)total/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

Output:

```
ANUSHKA _ 052
Enter number of process:3

Enter Burst Time:
p1:5
p2:8
p3:6

Process     Burst Time          Waiting Time    Turnaround Time
p1                5                 0                 5
p3                6                 5                 11
p2                8                 11                19

Average Waiting Time=5.333333
Average Turnaround Time=11.666667

Process returned 0 (0x0)   execution time : 4.297 s
Press any key to continue.
```

Experiment 6

Q. Write a program to implement shortest run time next algorithm.

Code:

```
#include<stdio.h>
int main()
{
 int at[10],bt[10],rt[10],endTime,i,smallest;
 int remain=0,n,time,sum_wait=0,sum_turnaround=0;
 printf("ANUSHKA _ 052\n");
 printf("Enter no of Processes : ");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
 printf("Enter arrival time for Process P%d : ",i+1);
 scanf("%d",&at[i]);
 printf("Enter burst time for Process P%d : ",i+1);
 scanf("%d",&bt[i]);
 rt[i]=bt[i];
 }
 printf("\n\nProcess\t|Turnaround Time| Waiting Time\n\n");
 rt[9]=9999;
 for(time=0;remain!=n;time++)
 {
 smallest=9;
 for(i=0;i<n;i++)
 {
 if(at[i]<=time && rt[i]<rt[smallest] && rt[i]>0)
 {
 smallest=i;
 }
 }
 rt[smallest]--;
 if(rt[smallest]==0)
 {
 remain++;
 endTime=time+1;
 printf("\nP[%d]\t|\t%d\t|\t%d",smallest+1,endTime-at[smallest],endTime-bt[smallest]-
at[smallest]);
 sum_wait+=endTime-bt[smallest]-at[smallest];
 sum_turnaround+=endTime-at[smallest];
 }
 }
 printf("\n\nAverage waiting time = %f\n",sum_wait*1.0/n);
 printf("Average Turnaround time = %f",sum_turnaround*1.0/5);
 return 0;
}
```

Output:

```
ANUSHKA _ 052
Enter no of Processes : 4
Enter arrival time for Process P1 : 3
Enter burst time for Process P1 : 4
Enter arrival time for Process P2 : 2
Enter burst time for Process P2 : 4
Enter arrival time for Process P3 : 6
Enter burst time for Process P3 : 5
Enter arrival time for Process P4 : 8
Enter burst time for Process P4 : 9


Process |Turnaround Time| Waiting Time


P[2]    |       4       |       0
P[1]    |       7       |       3
P[3]    |       9       |       4
P[4]    |       16      |       7

Average waiting time = 3.500000
Average Turnaround time = 7.200000
Process returned 0 (0x0)   execution time : 12.202 s
Press any key to continue.
```

Experiment 7

Q. Write a program in C to implement Priority Based Preemption algorithm.

Code:

```
#include<stdio.h>
int main()
{
 int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
 printf("ANUSHKA _ 052\n");
 printf("Enter Total Number of Process:");
 scanf("%d",&n);
 printf("\nEnter Burst Time and Priority\n");
 for(i=0;i<n;i++)
 {
 printf("\nP[%d]\n",i+1);
 printf("Burst Time:");
 scanf("%d",&bt[i]);
 printf("Priority:");
 scanf("%d",&pr[i]);
 p[i]=i+1;
 }
 for(i=0;i<n;i++)
 {
 pos=i;
 for(j=i+1;j<n;j++)
 {
 if(pr[j]<pr[pos])
 pos=j;
 }
 temp=pr[i];
 pr[i]=pr[pos];
 pr[pos]=temp;
 temp=bt[i];
 bt[i]=bt[pos];
 bt[pos]=temp;
 temp=p[i];
 p[i]=p[pos];
 p[pos]=temp;
 }
 wt[0]=0;
 for(i=1;i<n;i++)
 {
 wt[i]=0;
 for(j=0;j<i;j++)
 wt[i]+=bt[j];
 total+=wt[i];
 }
```

```
 avg_wt=total/n;
 total=0;
 printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
 for(i=0;i<n;i++)
 {
 tat[i]=bt[i]+wt[i];
 total+=tat[i];
 printf("\nP[%d]\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
 }
 avg_tat=total/n;
 printf("\n\nAverage Waiting Time=%d",avg_wt);
 printf("\nAverage Turnaround Time=%d\n",avg_tat);
return 0;
}
```

Output:

```
ANUSHKA _ 052
Enter Total Number of Process:4

Enter Burst Time and Priority

P[1]
Burst Time:5
Priority:8

P[2]
Burst Time:6
Priority:7

P[3]
Burst Time:3
Priority:6

P[4]
Burst Time:2
Priority:8

Process  Burst Time     Waiting Time    Turnaround Time
P[3]             3               0                      3
P[2]             6               3                      9
P[1]             5               9                      14
P[4]             2               14                     16

Average Waiting Time=6
Average Turnaround Time=10

Process returned 0 (0x0)   execution time : 15.408 s
Press any key to continue.
```

Experiment 8

Q. Write a program in C to implement Priority Based Non - Preemption algorithm.

Code:

```
#include <stdio.h>
int main()
{
 int pn = 0;
 int CPU = 0;
 int allTime = 0;
 printf("ANUSHKA _ 052\n");
 printf("Enrer Processes Count: ");
 scanf("%d",&pn);
 int AT[pn];
 int ATt[pn];
 int NoP = pn;
 int PT[pn];
 int PP[pn];
 int waittingTime[pn];
 int turnaroundTime[pn];
 for(int i=0 ;i<pn ;i++)
 {
 printf("\nProcessing time for P%d: ",i+1);
 scanf("%d",&PT[i]);
 printf("Piriorty for P%d: ",i+1);
 scanf("%d",&PP[i]);
 printf("Arrival Time for P%d: ",i+1);
 scanf("%d",&AT[i]);
 ATt[i] = AT[i];
 }
 int LAT = 0;
 for(int i = 0; i < pn; i++)
 if(AT[i] > LAT)
 LAT = AT[i];
 int ATv = AT[0];
 int ATi = 0;
 int P1 = PP[0];
 int P2 = PP[0];
 while(NoP > 0 && CPU <= 1000){
 for(int i = 0; i < pn; i++){
 if(ATt[i] < ATv){
 ATi = i;
 ATv = ATt[i];
 P1 = PP[i];
 P2 = PP[i];
 }
 else if(ATt[i] == ATv || ATt[i] <= CPU){
```

```
if(PP[i] != (pn+1))
P2 = PP[i];
if(P2 < P1){
ATi = i;
ATv = ATt[i];
P1 = PP[i];
P2 = PP[i];
}
}
}
if(CPU < ATv){
CPU = CPU+1;
continue;
}else{
waittingTime[ATi] = CPU - ATt[ATi];
CPU = CPU + PT[ATi];
turnaroundTime[ATi] = CPU - ATt[ATi];
ATt[ATi] = LAT +10;
ATv = LAT +10;
ATi = 0;
PP[ATi] = pn + 1;
P1 = PP[0];
P2 = PP[0];
printf("Iam in");
NoP = NoP - 1;
}
}
printf("\nPN\tPT\tPP\tWT\tTT\n\n");
for(int i = 0; i < pn; i++){
printf("P%d\t%d\t%d\t%d\t%d\n",i+1,PT[i],PP[i],waittingTime[i],turnaroundTime[i]);
}
int AvgWT = 0;
int AVGTaT = 0;
for(int i = 0; i < pn; i++){
AvgWT = waittingTime[i] + AvgWT;
AVGTaT = turnaroundTime[i] + AVGTaT;
}
printf("AvgWaittingTime = %d\nAvgTurnaroundTime = %d\n",AvgWT/pn,AVGTaT/pn);
return 0;
}
```

Output :

```
ANUSHKA _ 052
Enrer Processes Count: 4

Processing time for P1: 3
Piriorty for P1: 8
Arrival Time for P1: 1

Processing time for P2: 3
Piriorty for P2: 6
Arrival Time for P2: 8

Processing time for P3: 9
Piriorty for P3: 3
Arrival Time for P3: 7

Processing time for P4: 4
Piriorty for P4: 9
Arrival Time for P4: 6
Iam inIam inIam inIam in
PN      PT      PP      WT      TT

P1      3       5       0       3
P2      3       6       4199958 4200027
P3      9       3       1       10
P4      4       9       0       4
AvgWaittingTime = 1049989
AvgTurnaroundTime = 1050011

Process returned 0 (0x0)   execution time : 19.597 s
Press any key to continue.
```

Experiment 9

Q. Write a program in C to implement Round Robin algorithm.

Code:

```
#include<stdio.h>
int main()
{
 int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];
 float avg_wt, avg_tat;
 printf("ANUSHKA _ 052\n");
 printf("Total number of process in the system: ");
 scanf("%d", &NOP);
 y = NOP;
 for(i=0; i<NOP; i++)
 {
 printf("\nEnter the Arrival and Burst time of the Process[%d]", i+1);
 printf("\nArrival time is: \t");
 scanf("%d", &at[i]);
 printf("\nBurst time is: \t");
 scanf("%d", &bt[i]);
 temp[i] = bt[i];
 }
 printf("\nEnter the Time Quantum for the process: \t");
 scanf("%d", &quant);
 printf("\nProcess No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
 for(sum=0, i = 0; y!=0; )
 {
 if(temp[i] <= quant && temp[i] > 0)
 {
 sum = sum + temp[i];
 temp[i] = 0;
 count=1;
 }
 else if(temp[i] > 0)
 {
 temp[i] = temp[i] - quant;
 sum = sum + quant;
 }
 if(temp[i]==0 && count==1)
 {
 y--;
 printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);
 wt = wt+sum-at[i]-bt[i];
 tat = tat+sum-at[i];
 count =0;
 }
 if(i==NOP-1)
```

```
{
i=0;
}
else if(at[i+1]<=sum)
{
i++;
}
else
{
i=0;
}
}
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\nAverage Turn Around Time: \t%f", avg_wt);
printf("\nAverage Waiting Time: \t%f", avg_tat);
return 0;
}
```

Output:

```
ANUSHKA _ 052
Total number of process in the system: 3

Enter the Arrival and Burst time of the Process[1]
Arrival time is:        3

Burst time is:  5

Enter the Arrival and Burst time of the Process[2]
Arrival time is:        2

Burst time is:  3

Enter the Arrival and Burst time of the Process[3]
Arrival time is:        1

Burst time is:  1

Enter the Time Quantum for the process:         1

Process No              Burst Time              TAT             Waiting Time
Process No[3]           1                               3                       2
Process No[2]           3                               6                       3
Process No[1]           5                               6                       1
Average Turn Around Time:       2.000000
Average Waiting Time:   5.000000
Process returned 0 (0x0)   execution time : 13.726 s
Press any key to continue.
```

Experiment 10

Q. Write a program in C to implement Banker's algorithm.

Code:

```
#include <stdio.h>
int curr[5][5], maxclaim[5][5], avl[5];
int alloc[5] = {0, 0, 0, 0, 0};
int maxres[5], running[5], safe=0;
int count = 0, i, j, exec, r, p, k = 1;
int main()
{
 printf("ANUSHKA _ 052");
 printf("\nEnter the number of processes: ");
 scanf("%d", &p);
 for (i = 0; i < p; i++) {
 running[i] = 1;
 count++;
 }
 printf("\nEnter the number of resources: ");
 scanf("%d", &r);
 for (i = 0; i < r; i++) {
 printf("\nEnter the resource for instance%d: ", k++);
 scanf("%d", &maxres[i]);
 }
 printf("\nEnter maximum resource table:\n");
 for (i = 0; i < p; i++) {
 for(j = 0; j < r; j++) {
 scanf("%d", &maxclaim[i][j]);
 }
 }
 printf("\nEnter allocated resource table:\n");
 for (i = 0; i < p; i++) {
 for(j = 0; j < r; j++) {
 scanf("%d", &curr[i][j]);
 }
 }
 printf("\nThe resource of instances: ");
 for (i = 0; i < r; i++) {
 printf("\t%d", maxres[i]);
 }
 printf("\nThe allocated resource table:");
 for (i = 0; i < p; i++) {
 for (j = 0; j < r; j++) {
 printf("\t%d", curr[i][j]);
 }
 printf("\n");
 }
 printf("\nThe maximum resource table:");
```

```
for (i = 0; i < p; i++) {
for (j = 0; j < r; j++) {
printf("\t%d", maxclaim[i][j]);
}
printf("\n");
}
for (i = 0; i < p; i++) {
for (j = 0; j < r; j++) {
alloc[j] += curr[i][j];
}
}
printf("\nAllocated resources:\n");
for (i = 0; i < r; i++) {
printf("\t%d", alloc[i]);
}
for (i = 0; i < r; i++) {
avl[i] = maxres[i] - alloc[i];
}
printf("\nAvailable resources:\n");
for (i = 0; i < r; i++) {
printf("/t%d", avl[i]);
}
printf("\n");
while (count != 0) {
safe = 0;
for (i = 0; i < p; i++) {
if (running[i]) {
exec = 1;
for (j = 0; j < r; j++) {
if (maxclaim[i][j] - curr[i][j] > avl[j]) {
exec = 0;
break;
}
}
if (exec) {
printf("\nProcess%d is executing\n", i + 1);
running[i] = 0;
count--;
safe = 1;
for (j = 0; j < r; j++) {
avl[j] += curr[i][j];
}
break;
}
}
}
if (!safe) {
printf("\nThe processes are in unsafe state.");
break;
} else {
```

```
 printf("\nThe process is in safe state");
 printf("\nSafe sequence is:");
 for (i = 0; i < r; i++) {
 printf("t%d", avl[i]);
 }
 printf("\n");
 }
 }
}
```

Output:

```
ANUSHKA _ 052
Enter the number of processes: 4

Enter the number of resources: 4

Enter the resource for instance1: 8

Enter the resource for instance2: 5

Enter the resource for instance3: 9

Enter the resource for instance4: 7

Enter maximum resource table:
2 0 1 1
3 0 2 1
4 2 1 0
3 2 1 1

Enter allocated resource table:
3 0 6 1
4 5 0 2
3 4 7 1
0 3 2 6

The resource of instances:      8       5       9       7
The allocated resource table:   3       0       6       1
        4       5       0       2
        3       4       7       1
        0       3       2       6

The maximum resource table:     2       0       1       1
        3       0       2       1
        4       2       1       0
        3       2       1       1

Allocated resources:
        10      12      15      10
Available resources:
/t-2/t-7/t-6/t-3

The processes are in unsafe state.
Process returned 0 (0x0)   execution time : 107.614 s
Press any key to continue.
```

Experiment 11

Q. Write a program in C to implement Optimal Page Replacement algorithm.

Code :

```
#include<stdio.h>
int main()
{
 int no_of_frames, no_of_pages, frames[10], pages[30], temp[10],
 flag1, flag2, flag3, i, j, k, pos, max, faults = 0;
 printf("ANUSHKA _ 052\n");
 printf("Enter number of frames: ");
 scanf("%d", &no_of_frames);
 printf("Enter number of pages: ");
 scanf("%d", &no_of_pages);
 printf("Enter page reference string: ");
 for(i = 0; i < no_of_pages; ++i){
 scanf("%d", &pages[i]);
 }
 for(i = 0; i < no_of_frames; ++i){
 frames[i] = -1;
 }
 for(i = 0; i < no_of_pages; ++i){
 flag1 = flag2 = 0;
 for(j = 0; j < no_of_frames; ++j){
 if(frames[j] == pages[i]){
 flag1 = flag2 = 1;
 break;
 }
 }
 if(flag1 == 0){
 for(j = 0; j < no_of_frames; ++j){
 if(frames[j] == -1){
 faults++;
 frames[j] = pages[i];
 flag2 = 1;
 break;
 }
 }
 }
 if(flag2 == 0){
 flag3 =0;
 for(j = 0; j < no_of_frames; ++j){
 temp[j] = -1;
 for(k = i + 1; k < no_of_pages; ++k){
 if(frames[j] == pages[k])
 {
```

```
temp[j] = k;
break;
}
}
}
for(j = 0; j < no_of_frames; ++j){
if(temp[j] == -1){
pos = j;
flag3 = 1;
break;
}
}
if(flag3 ==0){
max = temp[0];
pos = 0;
for(j = 1; j < no_of_frames; ++j){
if(temp[j] > max){
max = temp[j];
pos = j;
}
}
}
frames[pos] = pages[i];
faults++;
}
printf("\n");
for(j = 0; j < no_of_frames; ++j){
printf("%d\t", frames[j]);
}
}
printf("\n\nTotal Page Faults = %d", faults);
return 0;
}
```

Output:

```
ANUSHKA _ 052
Enter number of frames: 4
Enter number of pages: 3
Enter page reference string: 2
2
3

2       -1      -1      -1
2       -1      -1      -1
2       3       -1      -1

Total Page Faults = 2
Process returned 0 (0x0)   execution time : 9.642 s
Press any key to continue.
```

Experiment 12

Q. Write a program in C to implement FIFO algorithm.

Code :

```
#include <stdio.h>
int main()
{
 int referenceString[10], pageFaults = 0, m, n, s, pages, frames;
 printf("ANUSHKA _ 052");
 printf("\nEnter the number of Pages:\t");
 scanf("%d", &pages);
 printf("\nEnter reference string values:\n");
 for( m = 0; m < pages; m++)
 {
 printf("Value No. [%d]:\t", m + 1);
 scanf("%d", &referenceString[m]);
 }
 printf("\n What are the total number of frames:\t");
 {
 scanf("%d", &frames);
 }
 int temp[frames];
 for(m = 0; m < frames; m++)
 {
 temp[m] = -1;
 }
 for(m = 0; m < pages; m++)
 {
 s = 0;
 for(n = 0; n < frames; n++)
 {
 if(referenceString[m] == temp[n])
 {
 s++;
 pageFaults--;
 }
 }
 pageFaults++;
 if((pageFaults <= frames) && (s == 0))
 {
 temp[m] = referenceString[m];
 }
 else if(s == 0)
 {
 temp[(pageFaults - 1) % frames] = referenceString[m];
 }
 printf("\n");
 for(n = 0; n < frames; n++)
```

```
 {
 printf("%d\t", temp[n]);
 }
 }
 printf("\nTotal Page Faults:\t%d\n", pageFaults);
 return 0;
}
```

Output:

```
ANUSHKA _ 052
Enter the number of Pages:      3

Enter reference string values:
Value No. [1]:  5
Value No. [2]:  3
Value No. [3]:  2

 What are the total number of frames:   4

5       -1      -1      -1
5       3       -1      -1
5       3       2       -1
Total Page Faults:      3

Process returned 0 (0x0)   execution time : 6.151 s
Press any key to continue.
```

Experiment 13

Q. Write a program in C to implement LRU algorithm.

Code :

```
#include<stdio.h>
int findLRU(int time[], int n)
{
 int i, minimum = time[0], pos = 0;
 for(i = 1; i < n; ++i){
 if(time[i] < minimum){
 minimum = time[i];
 pos = i;
 }
 }
 return pos;
}
int main()
{
 int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10],
 flag1, flag2, i, j, pos, faults = 0;
 printf("ANUSHKA _ 052\n");
 printf("Enter number of frames: ");
 scanf("%d", &no_of_frames);
 printf("Enter number of pages: ");
 scanf("%d", &no_of_pages);
 printf("Enter reference string: ");
 for(i = 0; i < no_of_pages; ++i){
 scanf("%d", &pages[i]);
 }
 for(i = 0; i < no_of_frames; ++i){
 frames[i] = -1;
 }
 for(i = 0; i < no_of_pages; ++i)
 {
 flag1 = flag2 = 0;
 for(j = 0; j < no_of_frames; ++j)
 {
 if(frames[j] == pages[i])
 {
 counter++;
 time[j] = counter;
 flag1 = flag2 = 1;
 break;
 }
 }
 if(flag1 == 0)
 {
```

```
for(j = 0; j < no_of_frames; ++j)
{
if(frames[j] == -1)
{
counter++;
faults++;
frames[j] = pages[i];
time[j] = counter;
flag2 = 1;
break;
}
}
}
if(flag2 == 0)
{
pos = findLRU(time, no_of_frames);
counter++;
faults++;
frames[pos] = pages[i];
time[pos] = counter;
}
printf("\n");
for(j = 0; j < no_of_frames; ++j)
{
printf("%d\t", frames[j]);
}
}
printf("\n\nTotal Page Faults = %d", faults);
return 0;
}
```

Output:

```
ANUSHKA _ 052
Enter number of frames: 3
Enter number of pages: 4
Enter reference string: 2
1
5
3

2       -1      -1
2       1       -1
2       1       5
3       1       5

Total Page Faults = 4
Process returned 0 (0x0)   execution time : 9.678 s
Press any key to continue.
```

Experiment 14

Q. Write a program in C to implement SCAN Disk Scheduling algorithm.

Code:

```
#include<stdio.h>
int main()
{
 int d[20],i,j,sum=0,n,disk,temp,max,dloc;
 printf("ANUSHKA _ 052\n");
 printf("enter number of location\t");
 scanf("%d",&n);
 printf("enter position of head\t");
 scanf("%d",&disk);
 printf("enter elements of disk queue\n");
 for(i=0;i<n;i++)
 {
 scanf("%d",&d[i]);
 }
 d[n]=disk;
 n=n+1;
 for(i=0;i<n;i++)
 {
 for(j=i;j<n;j++)
 {
 if(d[i]>d[j])
 {
 temp=d[i];
 d[i]=d[j];
 d[j]=temp;
 }
 }
 }
 max=d[n];
 for(i=0;i<n;i++)
 {
 if(disk==d[i])
 {
 dloc=i;
 break;
 }
 }
 for(i=dloc;i>=0;i--)
 {
 printf("%d -->",d[i]);
 }
 printf("0 -->");
```

```
 for(i=dloc+1;i<n;i++)
 {
 printf("%d-->",d[i]);
 }
 sum=disk+max;
 printf("\nmovement of total cylinders %d",sum);
 return 0;
}
```

Output:

```
ANUSHKA _ 052
enter number of location        3
enter position of head  4
enter elements of disk queue
1
2
3
4 -->3 -->2 -->1 -->0 -->
movement of total cylinders 1970236993
Process returned 0 (0x0)   execution time : 8.431 s
Press any key to continue.
```

Experiment 15

Q. Write a program in C to implement Shortest Seek Time First Disk Scheduling algorithm.

Code:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
 int RQ[100],i,n,TotalHeadMoment=0,initial,count=0;
 printf("ANUSHKA _ 052\n");
 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);
 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];
 RQ[index]=1000;
 count++;
 }
 printf("Total head movement is %d",TotalHeadMoment);
 return 0;
}
```

Output :

```
ANUSHKA _ 052
Enter the number of Requests
4
Enter the Requests sequence
2
1
2
3
Enter initial head position
4
Total head movement is 3
Process returned 0 (0x0)   execution time : 10.233 s
Press any key to continue.
```

Experiment 16

Q. Write a program in C to implement C-SCAN Disk Scheduling algorithm.

Code :

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
 int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
 printf("ANUSHKA _ 052\n");
 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);
 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(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];

}
}
else
{
for(i=index-1;i>=0;i--)
{
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];
}
TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0);
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:

```
ANUSHKA _ 052
Enter the number of Requests
3
Enter the Requests sequence
4
6
3
Enter initial head position
3
Enter total disk size
7
Enter the head movement direction for high 1 and for low 0
3
Total head movement is 11
Process returned 0 (0x0)   execution time : 10.757 s
Press any key to continue.
```