PART – A (SHEL SCRIPTS)

1. Write a shell program to find and display largest and smallestof threenumbers.

```
echo "enter the first number"
read a
echo " enter the second number"
read b
 echo "enter the third number "
  read c

  if [ $a -gt $b ] && [ $a -gt $c ]
    then
          echo " $a is greatest"

    elif [ $b -gt $a ] && [ $b -gt $c ]
  then
        echo "$b is greatest"
    else
          echo " $c is greatest "
  fi

  if [ $a -lt $b ] && [ $a -lt $c ]
    then
          echo " $a is smallest"

    elif [ $b -lt $a ] && [ $b -lt $c ]
  then
        echo "$b is smallest"
    else
          echo " $c is smallest "
   fi
```

2. Write a shell program to check the number n is divisible by m ornot Where m and n are supplied as command line argument or read from key board interactively.

Input read from key board :

```
#!/bin/sh
echo "enter value of n"
read n
echo "enter value of m"
read m
y=`expr $n % $m`
if [ $y -eq  0  ]
then
echo "$n is divisible by $m"
else
echo " $n is not divisible by $m "
fi
```

Input read through command line argument:

```
y=`expr $1 % $2`
if [ $y -eq 0 ] then
echo "$1 is divisible by $2"
else
echo " $1 is not divisible by $2 "
fi
```

3. Write a shell program to check the year is the leap year ornot. Displayappropriatemessage.

```
#!/bin/sh
echo "Enter year to check whether it is leap year or not"
read inpyr
x=`expr $inpyr % 4`
y=`expr $inpyr % 100`
z=`expr $inpyr % 400`
if [ $x -eq 0 ] && [ $y -ne 0 ] || [ $z -eq 0 ]
then
        echo "It is a leap year"
else
        echo "Not a leap year"
fi
```

4. Write a shell program that takes two file names ,checks the permission for these files are identical and if they are identical ,output the common permissions; otherwise output each file name followed by its permissions.

```
echo "enter two file names"
read f1 f2
if [ -e $f1 -a -e $f2 ]
then
        p1=`ls -l $f1 | cut -c 2-10`
        p2=`ls -l $f2 | cut -c 2-10`
        echo "$f1 :$p1"
        echo "$f2 :$p2"
        if [ "$p1"  =  "$p2"
        ]then
                echo "$f1 and $f2 have same permissions"
                echo "permission is: $p1"
        else
                echo "$f1 and $f2 have different permissions"

                echo "permissions for $f1:"
                echo"$p1"
                echo "permissions for$f2:"
                echo"$p2"
        fi
 else
        echo "invalid filename"
 fi
```

5. Write a shell program to display the length of the name and also display first three characters and last three characters in the name in two different lines if the name contains at least 6characters

```
echo "Enter your name:"
read name
if [ -z "$name" ]
then
        echo "NULL"
 else
        z=`expr "$name" : '.*' `
        echo "Length of string is:$z"

if [ $z -ge 6 ]
then
        echo "First 3 chars of name is:"
        m=`expr "$name" : '\(...\).*'`
        echo "$m"
        echo "Last 3 chars of name is:"
        n=`expr "$name" : '.*\(...\)'`
        echo "$n"
 fi
 fi
```

6. Write a shell program to implement simple calculatoroperations.

```
echo "Enter 2 operands and the operator"
read a
read b
 readop
case $op in
'+') res=`expr $a + $b`
echo "Sum is $res";;
'-') res=`expr $a - $b` echo
"Difference is$res";;
 '*') res=`expr $a \* $b`
echo "Product is $res";;
 '/') if [ $b - eq 0]
         then
         echo "division not possible"
         else
          res=`expr $a / $b`
          echo  "quotient=$res"

      fi
 ;;
'%') res=` expr $a % $b `
echo "res is $res";;
 *)echo "INVALID OPTION";;
 esac
```

7. Write a shell script that accepts filename as arguments. For every filename , it should first check whether it exists in the current directory and then convert its name to uppercase , but only if a file with new name doesn'texist.

```
for file in "$@"
do
if [ -f $file ]
then
  Ufile=`echo $file | tr '[a-z]' '[A-Z]'`
  if [ -f $Ufile ]
 then
echo "$Ufile also exists"
else
 mv $file $Ufile
fi
 else
echo "$file does not exist"
fi
 done
```

8. Write a shell script to determine the length of the string ,extract a substring and locate a position of a character in astring.

```
echo "enter the string"
read str
if [ -z $str ]
then
      echo "invalid string entered"
else
    y=`expr "$str" : '.*'`
    echo "length is : $y"
    if [ $y -ge 6 ]
    then
        z=`expr "$str" : '\(...\).*'`
        echo "$z"
    else
        echo "not possible"
    fi
fi
echo "enter the character to locate the position:"
read ch
if [ -z $ch ]
then
  echo "error!!!"
else
   r=`expr "$str" : '[^'$ch']*'$ch`
   echo "position: $r"
fi
```

9. Write a PERL program that prompts user to input the string and a number, and prints the string that many times, with each string on separateline.

```
#!/usr/bin/perl
print "\nenter the input string:";
$a= <STDIN>;
print "\nenter total number of times the string to be displayed: ";
chop ( $b = <STDIN> );

$c= $a x $b;
print "result is : \n$c ";
```

10. PERL program to find the sum of digits of an unsigned number passed through argument.

```
#!/usr/bin/perl
foreach $num (@ARGV)
 {
$sum=0;
$a=$num;
until($num==0)
  {
  $y=$num%10;
  $sum=$sum+$y;
$num=( $num /10);
}
print "sum of $a is :$sum\n"
 }
```

PART – B (C PROGRAMS)

1. Write the program to create five Child Process using system callfork() anddisplaytheirids.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
 int main(int argc,char* argv[])
 {
    int
    pid,i;for(i=0;i<5;i++)
    {
       pid=fork();
       if(pid==0)
       {
          printf("PID of process %d is = %d\n",i,getpid());
          exit(0);
       }
       wait(NULL);
    }
    return 0;
 }
```

2. Write a program to implement FCFS Scheduling algorithm to determine average wait time and average turn around time.

```
#include<stdio.h>
#include<stdlib.h>#incl
ude<unistd.h>
struct process
 {
int bt;
int wt;
int tt;
int pno;
 }p[10];
 int main()
 {
 int n,i;
float totwt=0,tott=0,avg1,avg2;
printf("enter no of process");
```

```
scanf("%d",&n);for(i=0;i<n;i++)
 {
printf("enter the bt of process%d",i+1);
scanf("%d",&p[i].bt);
 p[i].pno=i+1;
}p[0].wt=0;
p[0].tt=p[0].bt+p[0].wt;for(i=1;i
<n;i++)
 {
 p[i].wt=p[i-1].tt;
 p[i].tt=p[i].wt+p[i].bt;
 }
 for(i=0;i<n;i++)
 {
totwt=totwt+p[i].wt;tott=tott+p
[i].tt;
 }
printf("total wiating time:%f\n",totwt);
printf("total tt time:%f\n",tott);
avg1=(totwt/n);
```

```
printf("avgwt is:%f\n",avg1);
avg2=(tott/n);
printf("avgtt
is:%f\n",avg2);printf("process\tbt
\twt\ttt\n");for(i=0;i<n;i++)
 {
printf("p[%d]\t%d\t%d\t%d",(i+1), p[i].bt,p[i].wt,p[i].tt);
printf("\n");
 }
 return 0;
 }
```

3. **Write a program to implement SJF Scheduling algorithm todetermine average wait time and average turnaroundtime.**

```
#include<stdio.h>i
 nt main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_wt,avg_tat;
    printf("\nEnter number of process:");
    scanf("%d",&n);

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

   //sorting of burst times
    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]=te
        mp;
    }

    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     BurstTime     \tWaitingTime\tTurnaround
Time\n");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];total+=t
        at[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);
 }
```

4. Write the program to implement Round robin Scheduling algorithm to determine average wait time and average turnaroundtime.

```
#include<stdio.h>i

nt main()
 {

   int count,j,n,time,remain,flag=0,time_quantum;
   int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
   printf("Enter Total Process:\t ");
   scanf("%d",&n);
   remain=n;
   for(count=0;count<n;count++)
   {
    printf("Enter Arrival Time and Burst Time for Process Process
Number %d :",count+1);
     scanf("%d",&at[count]);
     scanf("%d",&bt[count]);rt[co
     unt]=bt[count];
   }
   printf("Enter Time Quantum:\t");
   scanf("%d",&time_quantum);
   printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
   for(time=0,count=0;remain!=0;)
   {
     if(rt[count]<=time_quantum && rt[count]>0)
     {
       time+=rt[count];rt[count]=
       0;
```

```
      flag=1;
    }
    else if(rt[count]>0)
    {
      rt[count]-
      =time_quantum;time+=ti
      me_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
      remain--;
     printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-
at[count]-bt[count]);
      wait_time+=time-at[count]-
      bt[count];turnaround_time+=time-
      at[count]; flag=0;
    }
    if(count==n-1)
      count=0;
    else if(at[count+1]<=time)
      count++;
    else
      count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
  printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

  return 0;
 }
```

5. Write a C program to simulate producer-consumer problem.

```
#include<stdio.h>
void main()
{
int buffer[10], bufsize, in, out, produce, consume, choice=0;in = 0;
out = 0;bufsize = 10;
while(choice !=3)
{
printf("\n1. Produce \t 2. Consume \t3. Exit");
printf("\nEnter your choice: ");
scanf("%d", &choice);
switch(choice)
{
case 1: if((in+1)%bufsize==out)
printf("\nBuffer is Full");
else
{
printf("\nEnter the value: ");
scanf("%d", &produce);
buffer[in] = produce;
in = (in+1)%bufsize;
}
break;
case 2:if(in == out)
printf("\nBuffer is Empty");
else
{
consume =buffer[out];
printf("\nThe consumed value is %d",consume); out
=(out+1)%bufsize;
}
```

```
break;
}
}
}
```

6. Write a program to demonstrate FIFO Page replacement algorithm to determine number of pagefaults.

```
#include<stdio.h>
#include<stdlib.h>#i
nclude<unistd.h>int
main()
 {
int noframes,noref,flag,i,j,frm[40],ref[40],pf=0,victim=-1;
printf("enter no 0f frames:");
scanf("%d",&noframes);
printf("enter the no ofreferences:");
scanf("%d",&noref); for(i=0;i<noref;i++)
 {
 scanf("%d",&ref[i]);
 }
 for(i=0;i<noframes;i++)
 {
 frm[i]=-1;
 }
printf("Reference String is:");
for(i=0;i<noref;i++)
 {
 printf("%d\t",ref[i]);
 }
 for(i=0;i<noref;i++)
 {
 flag=0;
printf("\n      %d-->",ref[i]);
for(j=0;j<noframes;j++)
```

```
 {
 if(frm[j]==ref[i])
 {
flag=1;
break;
 }
 }
 if(flag==0)
 {
pf++;
victim++;
victim=victim%noframes;frm[vi
ctim]=ref[i];
for(j=0;j<noframes;j++)
 {
 printf("%d\n\t",frm[j]);
 }
 printf("\t");
 }
 printf("\t");
 }
printf("\npage fault:%d",pf);
return 0;
 }
```

7. Write a program to demonstrate LRU Page replacement algorithm to determine number of pagefaults.

```
#include<stdio.h>
void print(int a[],intf){
printf("Frames:"); for(int
i=0;i<f;i++)
 {
 printf("%d ",a[i]);
 }
printf("\n");}
void main()
 {
 int n;
printf("Enter reference length:");
scanf("%d",&n);
 int a[n];
printf("Enter  reference string:");
for(int i=0;i<n;i++)
 {
 scanf("%d",&a[i]);
 }
 int f;
printf("Enter no. of free frames:");
scanf("%d",&f);
int free=f; int
frames[f]; int
count[f]; int
pos=0;
intpf=0;
 for(int j=0;j<f;j++)
 {
```

```
count[j]=0;
frames[j]=-1;
 }
 int time=0;
 for(int i=0;i<n;i++)
 {
int found=0;
for(intj=0;j<f;j++)
 {
 if(frames[j]==a[i])
 {
count[j]=time;f
ound=1; break;
 }
 }
 if(!found)
 {
 if(free)
 {
frames[pos]=a[i];pf
++;
free--
;count[pos]=time;p
os++;
 }
 else{
int min=count[0];
int index=0;
for(int j=1;j<f;j++)
if(count[j]<min)
 {
min=count[j];index
=j;
 }
frames[index]=a[i];c
ount[index]=time;
pf++;
```

```
 }
 }
time++;print(fram
es,f);
 }
 printf("\nNo of page faults:%d",pf);
 }
```

8. Write a program to demonstrate Optimal Page replacement algorithm to determine number of pagefaults.

```
#include<stdio.h>
#include<stdlib.h>#i
nclude<unistd.h>int
nor,nof;
int ref[20];
int fr[20];
 int optindex[10];
 int optimal(int index)
 {
 int pos;
 for(int i=0;i<nof;i++)
 {
 int notfound=1;
 for(int j=index+1;j<nor;j++)
 {
 if(fr[i]==ref[j])
 {
notfound=0;
optindex[i]=j;
break;
 }
 }
 if(notfound==1)
 {
 return i;
 }
 }
```

```
int max=optindex[0];
for(int i=0;i<nof;i++)
 {
 if(max<optindex[i])
 {
max=optindex[i];po
s=i;
 }
 }
 return pos;
 }
 int main()
 {
int fault;
intcount=0;
 int pf=0,victim=-1;
printf("enter number of pages referencingstring:");
scanf("%d",&nor);
printf("enter number offrames:");
scanf("%d",&nof);
printf("enter the reference string:");
for(int i=0;i<nor;i++)
 {
 scanf("%d",&ref[i]);
 }
 for(int i=0;i<nof;i++)
 {
optindex[i]=-
1;fr[i]=-1;
 }
 for(int i=0;i<nor;i++)
 {
 fault=1;
printf("\n%d ->",ref[i]);
for(int j=0;j<nof;j++)
 {
 if(ref[i]==fr[j])
 {
```

```
 fault=0;
 break;
  }
  }
  if(fault==1)
  {
  count++;

  if(count<=nof)
  {
  victim++;
  }
  else
  {
  victim=optimal(i);
  }
 pf++;
 fr[victim]=ref[i];
 for(int j=0;j<nof;j++)
  {
  printf("%4d",fr[j]);
  }
  }
  }
  printf("\n number of page fault:%d",pf);
  }
```

9. Write a program to demonstrate Bankers Deadlock avoidance algorithm.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 int n,m,i,j,k;
printf("Enter no. of processes:\n");
scanf("%d",&n);
printf("Enter no. of resources:\n");
 scanf("%d",&m);
 int alloc[n][m],max[n][m],avail[m],need[n][m];
printf("Enter the allocation matrix:");
```

```
for(i=0;i<n;i++)
     for(j=0;j<m;j++)
     scanf("%d",&alloc[i][j]);
printf("Enter the max matrix:");
    for(i=0;i<n;i++)
    for(j=0;j<m;j++)
scanf("%d",&max[i][j]);
printf("Enter the availble matrix:");
   for(i=0;i<m;i++)
scanf("%d",&avail[i]);
   for(i=0;i<n;i++)
   for(j=0;j<m;j++)
           need[i][j]=max[i][j]-alloc[i][j];
printf("NEED MATRIX IS:");
  for(i=0;i<n;i++)
  {
    for(j=0;j<m;j++)
printf("%d\t",need[i][j]);
printf("\n");
  }
   int safeseq[n],finish[n],found=0,count=0;
  for(i=0;i<n;i++)
   finish[i]=0;
   while(count<n)
    {
    found=0;
   for(i=0;i<n;i++)
  {
      if(finish[i]==0)
          {
       for(j=0;j<m;j++)
            {
             if(need[i][j]>avail[j])
                 break;
            }
    if(j==m)
        {
          for(k=0;k<m;k++)
                  avail[k]=avail[k]+alloc[i][k];
             safeseq[count++]=i;
               found=1;
                finish[i]=1;
```

```
            }
          }
      }
        if(found==0)
         {
    printf("SYSTEM IS IN UNSAFE STATE");
           return 0;
      }
     }
    printf("SYSTEM IS IN SAFE STATE");
    printf("Safe Sequence is:");
     for(k=0;k<n;k++)

    printf("%d\t",safeseq[k]+1);

}
```