PROGRAM CODING: 2

```
#include<stdio.h> #include<stdlib.h> main() {
FILE *fp; char c[100],pat[10];
int
l,i,j=0,count=0,len,k,flag=0; printf("\n enter the pattern");
scanf("%s",pat); len=strlen(pat); fp=fopen("nn.txt","r"); while(!feof(fp))

{ fscanf(fp,"%s",c);
l=strlen(c); count++;
for(i=0;i<count;i++)
{
if(c[i]==pat[j])
{ flag=0; for(k=1;k<i;k++)
{
if(c[i+k]!=pat[k]) flag=1; } if(flag==0)
printf("\n the pattern %s is present in word %d",pat,count); }}}}
```

PROGRAM CODING:

```
#!/bin/bash echo "enter the a vale" read a echo "enter b value" read b c=`expr $a + $b` echo "sum:"$c c=`expr $a - $b` echo "sub:"$c
c=`expr $a \* $b` echo "mul:"$c c=`expr $a / $b` echo "div:"$c
```

PROGRAM CODING:

```
#!/bin/bash num="1 2 3 4 5 6 7 8" for n in $num do q=`expr $n % 2`
if [ $q -eq 0 ] then echo "even no" continue
fi
echo "odd no" done
```

PROGRAM CODING:

```
#!/bin/bash
echo " which table you want" read n
for((i=1;i<10;i++))
do
echo $i "*" $n "=" `expr $i \* $n`
done
```

PROGRAM CODING:

```
#!/bin/bash a=1 while [ $a -lt 11 ] do
echo "$a" a=`expr $a + 1` done
```

PROGRAM CODING:

```
 #!/bin/bash
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 $a -gt $c ]
then echo "$a is greater" elif [ $b -gt $c ] then
echo "$b is greater" else
echo "$c is greater" fi
```

PROGRAM CODING:3

```
#include<stdio.h>
#include<stdlib.h> #include<unistd.h> void main(int argc,char *arg[])
{ int pid;

pid=fork(); if(pid<0)
{
printf("fork failed"); exit(1);
}
else if(pid==0)
```

```
{
execlp("whoami","ls",NULL); exit(0); } else {
printf("\n Process id is -d\n",getpid()); wait(NULL);
exit(0);
}
}
```

PROGRAM CODING:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h> int main( ) { int pid;
pid=fork( ); if(pid== 1)
{
 perror(“fork failed”); exit(0);
} if(pid==0) {
printf(“\n Child process is under execution”);
printf(“\n Process id of the child process is %d”, getpid()); printf(“\n Process id of the parent process is %d”, getppid());
} else {
printf(“\n Parent process is under execution”); printf(“\n Process id of the parent process is %d”, getpid()); printf(“\n Process id of the child process in parent is %d”, pid()); printf(“\n Process id of the parent of parent is %d”, getppid());
} return(0);
}
```

PROGRAM CODING:

```
#include<stdio.h>
#include<sys/types.h> #include<sys/dir.h>
void main(int age,char *argv[])
{ DIR *dir; struct dirent *rddir;
printf("\n Listing the directory content\n");
dir=opendir(argv[1]);
while((rddir=readdir(dir))!=NULL)
{
printf("%s\t\n",rddir->d_name);
}
closedir(dir);       }
```

PROGRAM CODING:

```
#include<stdio.h>
#include<sys/types.h> #include<sys/dir.h>
void main(int age,char *argv[])
{ DIR *dir; struct dirent *rddir;
printf("\n Listing the directory content\n");
dir=opendir(argv[1]);
while((rddir=readdir(dir))!=NULL)
{
printf("%s\t\n",rddir->d_name);
}
closedir(dir);
}
```

PROGRAM CODING:

```
 #include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<fcntl.h>
{ int fd[2]; char buf1[25]= ”just
O_RDWR); write(fd[0], Enter the text now….”);
SEEK_SET, 0);
write(fd[1], buf2, close(fd[1]); return0 }
```

PROGRAM :(FIRST COME FIRST SERVE SCHEDULING) 4

```
#include<stdio.h> struct process
{ int pid; int bt;
int wt,tt; }p[10]; int main()
{
int i,n,totwt,tottt,avg1,avg2; clrscr();
printf("enter the no of process \n"); scanf("%d",&n); for(i=1;i<=n;i++)
{ p[i].pid=i; printf("enter the burst time n"); scanf("%d",&p[i].bt);
} p[1].wt=0; p[1].tt=p[1].bt+p[1].wt; i=2; while(i<=n)
{
p[i].wt=p[i-1].bt+p[i-1].wt; p[i].tt=p[i].bt+p[i].wt; i ++; } i=1; totwt=tottt=0;
printf("\n processid \t bt\t wt\t tt\n"); while(i<=n){
printf("\n\t%d \t%d \t%d
\t%d",p[i].pid,p[i].bt,p[i].wt,p[i].tt); totwt=p[i].wt+totwt; tottt=p[i].tt+tottt; i++;}
avg1=totwt/n; avg2=tottt/n; printf("\navg1=%d \t avg2=%d\t",avg1,avg2
getch(); return 0;
}
```

PROGRAM: (PRIORITY SCHEDULING)

```
#include<stdio.h> #include<conio.h> struct process
{
int pid; int bt; int wt; int tt; int prior;
} p[10],temp; int main()
{
int i,j,n,totwt,tottt,arg1,arg2; clrscr();
printf("enter the number of process");
scanf("%d",&n); for(i=1;i<=n;i++)
{ p[i].pid=i;
printf("enter the burst time"); scanf("%d",&p[i].bt); printf("\n enter the priority"); scanf("%d",&p[i].prior);
}
for(i=1;i<n;i++)
{
for(j=i+1;j<=n;j++)
{ if(p[i].prior>p[j].prior)
{
 temp.pid=p[i].pid;
p[i].pid=p[j].pid; p[j].pid=temp.pid; temp.bt=p[i].bt; p[i].bt=p[j].bt; p[j].bt=temp.bt; temp.prior=p[i].prior; p[i].prior=p[j].prior;
p[j].prior=temp.prior;
}
}
} p[i].wt=0; p[1].tt=p[1].bt+p[1].wt; i=2; while(i<=n)
{
p[i].wt=p[i-1].bt+p[i-1].wt; p[i].tt=p[i].bt+p[i].wt; i++;
} i=1; totwt=tottt=0;
printf("\n process to \t bt \t wt \t tt");
while(i<=n)
{
printf("\n%d\t %d\t %d\t %d\t",p[i].pid,p[i].bt,p[i].wt,p[i].tt); totwt=p[i].wt+totwt; tottt=p[i].tt+tottt; i++; } arg1=totwt/n;
arg2=tottt/n;
printf("\n arg1=%d \t arg2=%d\t",arg1,arg2); getch(); return 0;
}
```

PROGRAM :(ROUND ROBIN SCHEDULING)

```
 #include<stdio.h> main() { int pt[10][10],a[10][10],at[10],pname[10][10],i,j,n,k=0,q,sum=0; float avg; printf("\n\n Enter the number
of processes : ");
scanf("%d",&n); for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
```

```
{ pt[i][j]=0; a[i][j]=0;
} }
for(i=0;i<n;i++)
{ j=0;
printf("\n\n Enter the process time for process %d : ",i+1); scanf("%d",&pt[i][j]);
}
printf("\n\n Enter the time slice : ");
scanf("%d",&q); printf("\n\n");
for(j=0;j<10;j++)
{
for(i=0;i<n;i++)
{ a[2*j][i]=k; if((pt[i][j]<=q)&&(pt[i][j]!=0))
{
pt[i][j+1]=0;
printf(" %d P%d %d\n",k,i+1,k+pt[i][j]); k+=pt[i][j]; a[2*j+1][i]=k;
}
else if(pt[i][j]!=0)
{ pt[i][j+1]=pt[i][j]-q;
printf(" %d P%d %d\n",k,i+1,(k+q)); k+=q; a[2*j+1][i]=k;
} else
{ a[2*j][i]=0; a[2*j+1][i]=0;
}
} } for(i=0;i<n;i++) sum+=a[0][i]; for(i=0;i<n;i++)
{
for(j=1;j<10;j++)
{
if((a[j][i]!=0)&&(a[j+1][i]!=0)&&((j+1)%2==0)) sum+=((a[j+1][i]-a[j][i]));
} }
avg=(float)sum/n;
printf("\n\n Average waiting time = %f msec",avg); sum=avg=0;
for(j=0;j<n;j++)
{ i=1;
while(a[i][j]!=0) i+=1
sum+=a[i-1][j];
}
avg=(float)sum/n; printf("\n\n Average turnaround time = %f msec\n\n",avg); }
```

PROGRAM: 5

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/shm.h> #include<sys/ipc.h>
int main()
{ int child,shmid,i; char * shmptr; child=fork(); if(!child)
{
shmid=shmget(2041,32,IPC_CREAT|0666);
shmptr=shmat(shmid,0,0); printf("\n Parent writing\n"); for(i=0;i<10;i++)
{ shmptr[i]="a"+i; putchar(shmptr[i]);
}
   }          else
{

 printf("\n\n %s", shmptr);  wait(NULL);
  }
shmid=shmget(2041,32,0666); shmptr=shmat(shmid,0,0); printf("\n Child is reading\n"); for(i=0;i<10;i++) putchar(shmptr[i]);
shmdt(NULL);
shmctl(shmid,IPC_RMID,NULL);
 return 0;
```

PROGRAM: (PRODUCER-CONSUMER PROBLEM) 6

```
#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;
            } }          }
```

PROGRAM: (SIMULATE ALGORITHM FOR DEADLOCK PREVENTION) 7

```
#include<stdio.h> #include<conio.h> void main() {
int k=0, output[10],d=0,t=0,ins[5],i,avail[5],allocated[10][5],need[10][5],MAX[10][5],pno,
P[10],j,rz, count=0;
clrscr();
printf("\n Enter the number of resources : ");
scanf("%d", &rz);
printf("\n enter the max instances of each resources\n"); for (i=0;i<rz;i++) { avail[i]=0; printf("%c= ",(i+97)); scanf("%d",&ins[i]);
}
printf("\n Enter the number of processes : "); scanf("%d", &pno);
printf("\n Enter the allocation matrix \n "); for (i=0;i<rz;i++) printf(" %c",(i+97));
printf("\n"); for (i=0;i <pno;i++) { P[i]=i; printf("P[%d] ",P[i]); for (j=0;j<rz;j++) { scanf("%d",&allocated[i][j]); avail[j]+=allocated[i][j];
} } printf("\nEnter the MAX matrix \n"); for (i=0;i<rz;i++) { printf(" %c",(i+97));
avail[i]=ins[i]-avail[i]; } printf("\n"); for (i=0;i <pno;i++) { printf("P[%d] ",i); for (j=0;j<rz;j++) scanf("%d", &MAX[i][j]);
} printf("\n"); A: d=-1; for (i=0;i <pno;i++) { count=0; t=P[i]; for (j=0;j<rz;j++) { need[t][j] = MAX[t][j]-allocated[t][j];
if(need[t][j]<=avail[j]) count++; } if(count==rz) { output[k++]=P[i]; for (j=0;j<rz;j++) avail[j]+=allocated[t][j];
} else
P[++d]=P[i]; } if(d!=-1) { pno=d+1; goto A; } printf("\t <"); for (i=0;i<k;i++) printf(" P[%d] ",output[i]); printf(">"); getch(); }
```

PROGRAM: 8

```
#include<stdio.h> #include<conio.h> int max[100][100]; int alloc[100][100]; int need[100][100]; int avail[100]; int n,r; void input();
void show(); void cal(); int main() {
int i,j;
printf("********** Deadlock Detection Algo ***********\n"); input();
show(); cal(); getch(); return 0; }
void input()
{int i,j;
printf("Enter the no of Processes\t");
scanf("%d",&n);
printf("Enter the no of resource instances
\t");
scanf("%d",&r);
printf("Enter the Max Matrix
\n");
for(i=0;i<n;i++) {for(j=0;j<r;j++) { scanf("%d",&max[i][j]);
}}
printf("Enter the Allocation Matrix
\n"); for(i=0;i<n;i++) {for(j=0;j<r;j++) { scanf("%d",&alloc[i][j]);
}}
printf("Enter the available Resources
```

```
\n"); for(j=0;j<r;j++) { scanf("%d",&avail[j]);
}} void show() {
int i,j;
printf("Process \t Allocation
\t Max
\t Available
\t");
for(i=0;i<n;i++) {
printf("
\nP%d
\t ",i+1); for(j=0;j<r;j++) { printf("%d ",alloc[i][j]); } printf("
\t"); for(j=0;j<r;j++) {printf("%d ",max[i][j]); } printf("
\t"); if(i==0) { for(j=0;j<r;j++) printf("%d ",avail[j]); }}} void cal()
{ int finish[100],temp,need[100][100],flag=1,k,c1=0; int dead[100];
int safe[100]; int i,j; for(i=0;i<n;i++)
{finish[i]=0;
}
//find need matrix for(i=0;i<n;i++) {for(j=0;j<r;j++)
{
need[i][j]=max[i][j]-alloc[i][j];
}} while(flag) {flag=0; for(i=0;i<n;i++) {int c=0; for(j=0;j<r;j++)
{if((finish[i]==0)&&(need[i][j]<=avail[j]))
{c++; if(c==r) { for(k=0;k<r;k++)
{avail[k]+=alloc[i][j]; finish[i]=1; flag=1; }//printf("\nP%d",i); if(finish[i]==1)
{i=n; }}}}}} j=0; flag=0; for(i=0;i<n;i++)
{ if(finish[i]==0) {dead[j]=i; j++; flag=1; }} if(flag==1) {
printf("\n\nSystem is in Deadlock and the Deadlock process are\n"); for(i=0;i<n;i++) {printf("P%d\t",dead[i]);
}} else {
printf("\nNo Deadlock Occur"); }}
```

PROGRAM: 9

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h> #include<unistd.h>
pthread_t tid[2];
void* doSomeThing(void *arg)
{ unsigned long i = 0; pthread_t id = pthread_self(); if(pthread_equal(id,tid[0]))
{
printf("\n First thread processing\n");
} else {
printf("\n Second thread processing\n");
} for(i=0; i<(0xFFFFFFFF);i++); return NULL;
}
int main(void)
{ int i = 0;

int err;
while(i < 2)
{
err = pthread_create(&(tid[i]), NULL, &doSomeThing, NULL); if (err != 0)
printf("\ncan't create thread :[%s]", strerror(err)); else
printf("\n Thread created successfully\n");

i++; } sleep(5); return 0;
}
```

PROGRAM: 10

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h> #include<unistd.h> pthread_t tid[2]; int counter; pthread_mutex_t lock; void* doSomeThing(void *arg)
{
pthread_mutex_lock(&lock); unsigned long i = 0; counter += 1;
printf("\n Job %d started\n", counter); for(i=0; i<(0xFFFFFFFF);i++); printf("\n Job %d finished\n", counter);
pthread_mutex_unlock(&lock); return NULL;
}
int main(void)
{ int i = 0; int err;
if (pthread_mutex_init(&lock, NULL) != 0)
{ printf("\n mutex init failed\n"); return 1; }
while(i < 2)
{
err = pthread_create(&(tid[i]), NULL, &doSomeThing, NULL); if (err != 0)
printf("\ncan't create thread :[%s]", strerror(err));
i++; }
pthread_join(tid[0], NULL); pthread_join(tid[1], NULL); pthread_mutex_destroy(&lock); return 0;}
```

PROGRAM 11

WORST-FIT

```
#include<stdio.h>
#include<conio.h> #define max 25 void main()
{ int
frag[max],b[max],f[max],i,j,nb,nf,t emp; static int bf[max],ff[max]; clrscr();
printf("\n\tMemory Management Scheme - First Fit");
printf("\nEnter the number of blocks:"); scanf("%d",&nb); printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n"); for(i=1;i<=nb;i++)
{
printf("Block %d:",i); scanf("%d",&b[i]);
} printf("Enter the size of the files :-\n"); for(i=1;i<=nf;i++)
{ printf("File %d:",i); scanf("%d",&f[i]);
} for(i=1;i<=nf;i++) { for(j=1;j<=nb;j++)
{ if(bf[j]!=1)
{ temp=b[j]-f[i]; if(temp>=0)
{
ff[i]=j;
break;
}

}
} frag[i]=temp; bf[ff[i]]=1;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement"); for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]); getch();
}
```

BEST FIT:

```
#include<stdio.h>
#include<conio.h> #define max 25 void main()
{ int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000; static int bf[max],ff[max];
clrscr();
printf("\nEnter the number of blocks:"); scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
```

```
printf("\nEnter the size of the blocks:-\n"); for(i=1;i<=nb;i++) printf("Block %d:",i); scanf("%d",&b[i]);
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{ printf("File %d:",i); scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{ for(j=1;j<=nb;j++)
{ if(bf[j]!=1)
{ temp=b[j]-f[i]; if(temp>=0) if(lowest>temp)
{
ff[i]=j;
lowest=temp;
} }} frag[i]=lowest; bf[ff[i]]=1; lowest=10000;
} printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment"); for(i=1;i<=nf && ff[i]!=0;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]); getch();
}
```

FIRSTFIT

```
 #include<stdio.h>
#include<conio.h> #define max 25 void main()
{ int
frag[max],b[max],f[max],i,j,nb,nf,temp,highes t=0; static int bf[max],ff[max]; clrscr();
printf("\n\tMemory Management Scheme - Worst Fit");
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n"); for(i=1;i<=nb;i++)
{ printf("Block %d:",i); scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{ printf("File %d:",i); scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{ for(j=1;j<=nb;j++)
{ if(bf[j]!=1) //if bf[j] is not allocated
{ temp=b[j]-f[i]; if(temp>=0) if(highest<temp)
{
}
}
frag[i]=highest; bf[ff[i]]=1; highest=0;
} ff[i]=j; highest=temp;
printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragement"); for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
getch();}
```

PROGRAM: (SIMULATE PAGE REPLACEMENT ALGORITHMS FIFO) 12

```
#include<stdio.h> #include<conio.h>
int i,j,nof,nor,flag=0,ref[50],frm[50],pf=0,victim=-1; void main()
{ clrscr();
printf("\n \t\t\t FIFI PAGE REPLACEMENT ALGORITHM"); printf("\n Enter no.of frames. .. "); scanf("%d",&nof);
printf("Enter number of Pages.\n");
scanf("%d",&nor);
printf("\n Enter the Page No. . "); for(i=0;i<nor;i++) scanf("%d",&ref[i]); printf("\nThe given Pages are:"); for(i=0;i<nor;i++)
printf("%4d",ref[i]); for(i=1;i<=nof;i++) frm[i]=-1; printf("\n"); for(i=0;i<nor;i++)
{ flag=0;
printf("\n\t page no %d->\t",ref[i]);
```

```
for(j=0;j<nof;j++)
{
if(frm[j]==ref[i])
{ flag=1; break; }} if(flag==0)
{
pf++; victim++; victim=victim%nof; frm[victim]=ref[i]; for(j=0;j<nof;j++) printf("%4d",frm[j]);
} } printf("\n\n\t\t No.of pages faults. . %d",pf); getch();
}
```

PROGRAM: (SIMULATE PAGE REPLACEMENT ALGORITHMS: LRU)

```
#include<stdio.h> #include<conio.h>
int i,j,nof,nor,flag=0,ref[50],frm[50],pf=0,victim=-1; int recent[10],lrucal[50],count=0;
int lruvictim(); void main()
{ clrscr();
printf("\n\t\t\t LRU PAGE REPLACEMENT ALGORITHM"); printf("\n Enter no.of Frames... "); scanf("%d",&nof);

printf(" Enter no.of reference string.."); scanf("%d",&nor);
printf("\n Enter reference string..");
for(i=0;i<nor;i++) scanf("%d",&ref[i]);
printf("\n\n\t\t LRU PAGE REPLACEMENT ALGORITHM ");
printf("\n\t The given reference string:"); printf("\n… ..... "); for(i=0;i<nor;i++) printf("%4d",ref[i]);
for(i=1;i<=nof;i++)
{ frm[i]=-1; lrucal[i]=0; }
for(i=0;i<10;i++)
recent[i]=0; printf("\n"); for(i=0;i<nor;i++)
{ flag=0;
printf("\n\t Reference NO %d->\t",ref[i]); for(j=0;j<nof;j++)
{
if(frm[j]==ref[i])
{ flag=1; break;
} } if(flag==0)
{
count++; if(count<=nof) victim++;
else victim=lruvictim(); pf++; frm[victim]=ref[i]; for(j=0;j<nof;j++) printf("%4d",frm[j]);

recent[ref[i]]=i;
}
printf("\n\n\t No.of page faults...%d",pf); getch(); }
int lruvictim()
{ int i,j,temp1,temp2; for(i=0;i<nof;i++)
{ temp1=frm[i]; lrucal[i]=recent[temp1];
} temp2=lrucal[0]; for(j=1;j<nof;j++)
{
if(temp2>lrucal[j]) temp2=lrucal[j];
} for(i=0;i<nof;i++) if(ref[temp2]==frm[i]) return i; return 0;
}
```

PROGRAM: (SIMULATE PAGE REPLACEMENT ALGORITHMS: OPTIMAL)

```
#include<stdio.h> #include<conio.h> int n,page[20],f,fr[20],i; void display()
{
for(i=0;i<f;i++)
{
printf("%d",fr[i]);
}
printf("\n");
}
void request()
{
printf("enter no.of pages:"); scanf("%d",&n);
```

```
printf("enter no.of frames:"); scanf("%d",&f);
printf("enter no.of page no.s"); for(i=0;i<n;i++)
{
scanf("%d",&page[i]);
}
for(i=0;i<n;i++)
{
fr[i]=-1;
} }
void replace()
{
int j,flag=0,pf=0; int max,lp[10],index,m; for(j=0;j<f;j++)
{
fr[j]=page[j]; flag=1; pf++; display();
}
for(j=f;j<n;j++)
{ flag=0;
for(i=0;i<f;i++)
{
if(fr[i]==page[j])
{ flag=1; break;
} }
if(flag==0)
{
for(i=0;i<f;i++) lp[i]=0;
for(i=0;i<f;i++)
{
for(m=j+1;m<n;m++)
{
if(fr[i]==page[m])
{ lp[i]=m-j; break;
}
}
} max=lp[0]; index=0; for(i=0;i<f;i++)
{
if(lp[i]==0)
{
index=i; break; } else {
if(max<lp[i])
{ max=lp[i]; index=i;
}
} }
fr[index]=page[j]; pf++; display();
} }
printf("page faults:%d",pf);
} void main()
{ clrscr(); request(); replace(); getch();
}
```

PROGRAM: 13

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h> #include<graphics.h> void main()
{
int gd=DETECT,gm,count,i=0,j,mid,cir_x; char fname[10][20]; clrscr();
initgraph(&gd,&gm,"c:\\Turboc3\\BGI"); cleardevice(); setbkcolor(BLUE); printf("enter number of files:"); scanf("%d",&count);
if(i<count)
{ cleardevice(); setbkcolor(6);
```

```
printf("enter %d file name:",i+1); scanf("%s",fname[i]); setfillstyle(1,MAGENTA); mid=640/count; cir_x=mid/3;
bar3d(270,100,370,150,0,0);
settextstyle(2,0,4); settextjustify(1,1);
outtextxy(320,125,"root directory"); setcolor(10); i++;
for(j=0;j<i;j++,cir_x+=mid)
{ line(320,150,cir_x,250); fillellipse(cir_x,250,30,30); outtextxy(cir_x,250,fname[j]);
} } getch(); closegraph();
}
```

PROGRAM:

```
 #include<stdio.h>
#include<conio.h> #include<graphics.h> struct tree_element
{
char name[20];
int x,y,ftype,lx,rx,nc,level; struct tree_element *link[5];
};
typedef struct tree_element node; void main()
{
int gd=DETECT,gm;
node *root; root=NULL;
clrscr();
create(&root,0,"null",0,639,320); clrscr();
initgraph(&gd,&gm,"c:\\Turboc3\\BGI"); display(root); getch(); closegraph();
}
create(node **root,int lev ,char *dname,int lx,int rx,int x)
{ int i, gap;
if(*root==NULL)
{
(*root)=(node*)malloc(sizeof(node));
printf("\nEnter the name of the file name %s:",dname); fflush(stdin);
```

PROGRAM:

```
 #include<stdio.h>
#include<conio.h> #include<graphics.h> struct tree_element
{
char name[20];
int x,y,ftype,lx,rx,nc,level; struct tree_element *link[5];
};
typedef struct tree_element node; void main()
{
int gd=DETECT,gm;
node *root; root=NULL;
clrscr();
create(&root,0,"null",0,639,320); clrscr();
initgraph(&gd,&gm,"c:\\Turboc3\\BGI"); display(root); getch(); closegraph();
}
create(node **root,int lev ,char *dname,int lx,int rx,int x)
{ int i, gap;
if(*root==NULL)
{
(*root)=(node*)malloc(sizeof(node));
printf("\nEnter the name of the file name %s:",dname); fflush(stdin);
gets((*root)->name); if(lev==0 || lev==1) (*root)-> ftype=1; else
(*root)->ftype=2;
(*root)->level=lev;
(*root)->y=50+lev*50;
```

```
(*root)->x=x;
(*root)->lx=lx ; (*root)->rx=rx; for(i=0;i<5;i++) (*root)->link[i]=NULL; if((*root)->ftype==1)
{
if(lev==0 || lev==1)
{
if((*root)->level==0) printf("\nHow many users:"); else
printf("\nHow many files:"); printf("(for %s):",(*root)->name); scanf("%d",&(*root)->nc);
} else
(*root)->nc=0; if((*root)->nc==0) gap=rx-lx; else
gap=(rx-lx)/(*root)->nc; for(i=0;i<(*root)->nc;i++)
create(&((*root)->link[i]),lev+1,(*root)->name,lx+gap*i,lx+gap*i+gap,lx+gap*i+gap/2);
} else
(*root)->nc=0;
} }
display(node *root)
{
int i; settextstyle(2,0,4); settextjustify(1,1); setfillstyle(1,BLUE); setcolor(14);
if(root!=NULL)
{
for(i=0;i<root->nc;i++)
{
line(root->x,root->y,root->link[i]->x,root->link[i]->y);
}
if(root->ftype==1) bar3d(root->x-20, root->y-10,root->x+20,root->y+10,0,0);
else fillellipse(root->x,root->y,20,20); outtextxy(root->x,root->y,root->name); for(i=0;i<root->nc;i++)
{
display(root->link[i]);
}
}
}
```

PROGRAM:

```
 #include<stdio.h>
#include<conio.h>
#include<string.h> #include<graphics.h> struct tree_element
{ char name[20]; int x,y,ftype,lx,rx,nc,level; struct tree_element *link[5];
};
typedef struct tree_element node; typedef struct
{ char from[20]; char to[20]; }link; link L[10]; int nofl; node *root; void main()
{
int gd=DETECT,gm;
root=NULL;
clrscr();
create(&root,0,"root",0,639,320); read_links();
clrscr();
initgraph(&gd,&gm,"c:\\Turboc3\\BGI");
draw_link_lines(); display(root); getch(); closegraph();
}
create(node **root,int lev,char *dname,int lx,int rx,int x)
{
int i,gap;
if(*root==NULL)
{
(*root)=(node*)malloc(sizeof(node)); printf("\nEnter name of dir/file(under %s):",dname); fflush(stdin); gets((*root)->name);
printf("\nEnter 1 for Dir/2 for File:"); scanf("%d",&(*root)->ftype);
(*root)->level=lev;
(*root)->y=50+lev*50;
(*root)->x=x;
```

```
(*root)->lx=lx; (*root)->rx=rx; for(i=0;i<5;i++) (*root)->link[i]=NULL; if((*root)->ftype==1)
{
printf("\nNo of Sub Directories/Files(for %s):",(*root)->name); scanf("%d",&(*root)->nc); if((*root)->nc==0) gap=rx-lx; else gap=(rx-
lx)/(*root)->nc; for(i=0;i<(*root)->nc;i++)
create(&((*root)->link[i]),lev+1,(*root)->name,lx+gap*i,lx+gap*i+gap,lx+gap*i+gap/2);
} else
(*root)->nc=0;
} }
read_links()
{
int i;
printf("\nHow many Links:"); scanf("%d",&nofl); for(i=0;i<nofl;i++)
{
printf("\nFile/Dir:"); fflush(stdin); gets(L[i].from); printf("\nUser Name:");
fflush(stdin); gets(L[i].to);
} }
draw_link_lines()
{ int i,x1,y1,x2,y2; for(i=0;i<nofl;i++)
{ search(root,L[i].from,&x1,&y1);
search(root,L[i].to,&x2,&y2);
setcolor(LIGHTGREEN);
setlinestyle(3,0,1);
line(x1,y1,x2,y2); setcolor(YELLOW);
setlinestyle(0,0,1);
} }
search(node *root,char *s,int *x,int *y)
{
int i;
if(root!=NULL)
{
if(strcmpi(root->name,s)==0)
{
*x=root->x;
*y=root->y; return; } else { for(i=0;i<root->nc;i++) search(root->link[i],s,x,y);
}}}
display(node *root)
{
int i;
settextstyle(2,0,4); settextjustify(1,1); setfillstyle(1,BLUE); setcolor(14);
if(root!=NULL)
{ for(i=0;i<root->nc;i++)
{
line(root->x,root->y,root->link[i]->x,root->link[i]->y);
}
if(root->ftype==1)
bar3d(root->x-20,root->y-10,root->x+20,root->y+10,0,0); else
fillellipse(root->x,root->y,20,20); outtextxy(root->x,root->y,root->name); for(i=0;i<root->nc;i++)
{
display(root->link[i]); }}}
PROGRAM:14
#include<stdio.h> #include<conio.h> main()
{ int n,i,j,b[20],sb[20],t[20],x,c[20][20]; clrscr(); printf("Enter no.of files:"); scanf("%d",&n); for(i=0;i<n;i++)
{
printf("Enter no. of blocks occupied by file%d",i+1); scanf("%d",&b[i]);
printf("Enter the starting block of file%d",i+1);
scanf("%d",&sb[i]); t[i]=sb[i]; for(j=0;j<b[i];j++) c[i][j]=sb[i]++; } printf("Filename\tStart block\tlength\n"); for(i=0;i<n;i++) printf("%d\t
%d \t%d\n",i+1,t[i],b[i]); printf("Enter file name:"); scanf("%d",&x); printf("File name is:%d",x); printf("length is:%d",b[x-1]);
printf("blocks occupied:"); for(i=0;i<b[x-1];i++) printf("%4d",c[x-1][i]); getch();
```

```
}
```
PROGRAM:
```
#include<stdio.h> #include<conio.h> main()
{ int n,m[20],i,j,sb[20],s[20],b[20][20],x; clrscr(); printf("Enter no. of files:"); scanf("%d",&n); for(i=0;i<n;i++)
{ printf("Enter starting block and size of file%d:",i+1); scanf("%d%d",&sb[i],&s[i]); printf("Enter blocks occupied by
file%d:",i+1); scanf("%d",&m[i]); printf("enter blocks of file%d:",i+1); for(j=0;j<m[i];j++) scanf("%d",&b[i][j]);
} printf("\nFile\t index\tlength\n"); for(i=0;i<n;i++)
{ printf("%d\t%d\t%d\n",i+1,sb[i],m[i]);
}printf("\nEnter file name:"); scanf("%d",&x);
printf("file name is:%d\n",x); i=x-1; printf("Index is:%d",sb[i]); printf("Block occupied are:"); for(j=0;j<m[i];j++) printf("%3d",b[i][j]);
getch();
}
```
PROGRAM:
```
#include<stdio.h> #include<conio.h>
struct file
{ char fname[10]; int start,size,block[10];
}f[10]; main()
{ int i,j,n; clrscr(); printf("Enter no. of files:"); scanf("%d",&n); for(i=0;i<n;i++)
{ printf("Enter file name:"); scanf("%s",&f[i].fname); printf("Enter starting block:");
scanf("%d",&f[i].start);
f[i].block[0]=f[i].start; printf("Enter no.of blocks:"); scanf("%d",&f[i].size); printf("Enter block numbers:"); for(j=1;j<=f[i].size;j++)
{ scanf("%d",&f[i].block[j]);
} } printf("File\tstart\tsize\tblock\n"); for(i=0;i<n;i++)
{ printf("%s\t%d\t%d\t",f[i].fname,f[i].start,f[i].size); for(j=1;j<=f[i].size-1;j++) printf("%d--->",f[i].block[j]);
printf("%d",f[i].block[j]); printf("\n");
} getch();
}
```

PROGRAM 15

A) FCFS DISK SCHEDULING ALGORITHM
```
#include<stdio.h> main()
{ int t[20], n, I, j, tohm[20], tot=0; float avhm; clrscr();
printf(“enter the no.of tracks”); scanf(“%d”,&n);
printf(“enter the tracks to be traversed”); for(i=2;i<n+2;i++) scanf(“%d”,&t*i+); for(i=1;i<n+1;i++)
{ tohm[i]=t[i+1]-t[i]; if(tohm[i]<0)
tohm[i]=tohm[i]*(-1);
}
for(i=1;i<n+1;i++) tot+=tohm[i]; avhm=(float)tot/n;
printf(“Tracks traversed\tDifference between tracks\n”); for(i=1;i<n+1;i++)
printf(“%d\t\t\t%d\n”,t*i+,tohm*i+); printf("\nAverage header movements:%f",avhm); getch();
}
```
B) SCAN DISK SCHEDULING ALGORITHM
```
#include<stdio.h> main()
{ int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0; clrscr();
printf("enter the no of tracks to be traveresed"); scanf("%d'",&n);
printf("enter the position of head");
scanf("%d",&h); t[0]=0;t[1]=h;
printf("enter the tracks"); for(i=2;i<n+2;i++) scanf("%d",&t[i]); for(i=0;i<n+2;i++)
{ for(j=0;j<(n+2)-i-1;j++)
{ if(t[j]>t[j+1])
{ temp=t[j]; t[j]=t[j+1]; t[j+1]=temp;
}
}
}
for(i=0;i<n+2;i++) if(t[i]==h)
j=i;k=i; p=0;
while(t[j]!=0)
{
```

```
atr[p]=t[j]; j--; p++;
}
atr[p]=t[j];
for(p=k+1;p<n+2;p++,k++) atr[p]=t[k+1]; for(j=0;j<n+1;j++)
{ if(atr[j]>atr[j+1]) d[j]=atr[j]-atr[j+1];
else
d[j]=atr[j+1]-atr[j]; sum+=d[j];
}
printf("\nAverage header movements:%f",(float)sum/n); getch(); }
```

C) C-SCAN DISK SCHEDULING ALGORITHM

```
#include<stdio.h> main()
{ int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0; clrscr();
printf("enter the no of tracks to be traveresed"); scanf("%d'",&n);
printf("enter the position of head");
scanf("%d",&h); t[0]=0;t[1]=h;
printf("enter total tracks"); scanf("%d",&tot);
t[2]=tot-1; printf("enter the tracks"); for(i=3;i<=n+2;i++) scanf("%d",&t[i]); for(i=0;i<=n+2;i++)
for(j=0;j<=(n+2)-i-1;j++)
if(t[j]>t[j+1])
{ temp=t[j]; t[j]=t[j+1]; t[j+1]=temp }
for(i=0;i<=n+2;i++) if(t[i]==h); j=i;break; p=0;
while(t[j]!=tot-1)
{ atr[p]=t[j]; j++; p++;
}
atr[p]=t[j]; p++; i=0; while(p!=(n+3) && t[i]!=t[h])
{
atr[p]=t[i]; i++;
p++;
}
for(j=0;j<n+2;j++)
{
if(atr[j]>atr[j+1])
{ d[j]=atr[j]-atr[j+1];
```