CS/FCFS:

```
#include<stdio.h>
int main()
{
int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp; float
avg_wt,avg_tat;
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;
//contains process number
}
wt[0]=0;
//waiting time for first process will be zero
//calculate waiting time
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;
//average waiting time
total=0;
printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];
//calculate turnaround time
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;
//average turnaround time
printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

CS/SJF

```
#include<stdio.h>
int main()
{
int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp; float
avg_wt,avg_tat;
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; }
//sorting burst time in ascending order using selection sort
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;
//waiting time for first process will be zero
//calculate waiting time
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;
//average waiting time
total=0;
printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];
//calculate turnaround time
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;
//average turnaround time
printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

CS/PS

```
#include<stdio.h>
int main()
{
int bt[20],p[20],wt[20],tat[20],pri[20],i,j,k,n,total=0,pos,temp; float
avg_wt,avg_tat;
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;
//contains process number}
printf(" enter priority of the process ");
for(i=0;i<n;i++){
p[i] = i;
//printf("Priority of Process");
printf("p%d ",i+1);scanf("%d",&pri[i]);}
for(i=0;i<n;i++)
for(k=i+1;k<n;k++)
if(pri[i] > pri[k]){
temp=p[i];
p[i]=p[k];
p[k]=temp;
temp=bt[i];
bt[i]=bt[k];
bt[k]=temp;
temp=pri[i];
pri[i]=pri[k];
pri[k]=temp;
}wt[0]=0;
//waiting time for first process will be zero
//calculate waiting time
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;
//average waiting time
total=0;
printf("\nProcess\t
Burst Time \tPriority \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++){
tat[i]=bt[i]+wt[i];
//calculate turnaround time
total+=tat[i];
printf("\np%d\t\t %d\t\t %d\t\t %d\t\t\t\t%d",p[i],bt[i],pri[i],wt[i],tat[i]);}
avg_tat=(float)total/n;
//average turnaround time
printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\nAverage Turnaround Time=%f\n",avg_tat);}
```

CS/RR

```
#include<stdio.h>
main()
{
int st[10],bt[10],wt[10],tat[10],n,tq; int
i,count=0,swt=0,stat=0,temp,sq=0;
float awt,atat;
printf("enter the number of processes");
scanf("%d",&n);
printf("enter the burst time of each process /n");
for(i=0;i<n;i++)
{
printf(("p%d",i+1);
scanf("%d",&bt[i]);
st[i]=bt[i];
}
printf("enter the time quantum");
scanf("%d",&tq);
while(1)
{
for(i=0,count=0;i<n;i++)
{
temp=tq;
if(st[i]==0)
{
count++;
continue;
}
if(st[i]>tq)
st[i]=st[i]-tq;
else
if(st[i]>=0){
temp=st[i];
st[i]=0;
}
sq=sq+temp;
tat[i]=sq;
}
if(n==count)
break;
}
for(i=0;i<n;i++)
{
wt[i]=tat[i]-bt[i];
swt=swt+wt[i];
stat=stat+tat[i];
}
awt=(float)swt/n;
atat=(float)stat/n;
printf("process no\t burst time\t waiting time\t turnaround time\n");
for(i=0;i<n;i++)
printf("%d\t\t %d\t\t %d\t\t %d\n",i+1,bt[i],wt[i],tat[i]); printf("avg
wt time=%f,avg turn around time=%f",awt,atat);
}
```

IPC/PIPE

```
#include <stdio.h>
int main()
{
int fd[2],child; char a[10];
printf("\n Enter the string:");
scanf("%s",a);
pipe(fd);
child=fork();
if(!child)
{
close(fd[0]);
write(fd[1],a,5); wait(0);
}
else
{
close(fd[1]);read(fd[0],a,5);
printf("The string received from pipe is: %s",a);
}
return 0;
}
```

IPC/SYSTEM CALL READ WRITE CREATEFORKOPENCLOSE

```
#include<sys/stat.h>
#include<stdio.h>
#include<fcntl.h>
#include<sys/types.h>
int main()
{
int n,i=0;
int f1,f2;
char c,strin[100];
f1=open("data",O_RDWR|O_CREAT|O_TRUNC);
while((c=getchar())!='\n')
{
strin[i++]=c;
}
strin[i]='\0';
write(f1,strin,i);
close(f1);
f2=open("data",O_RDONLY);
read(f2,strin,0);
printf("\n%s\n",strin);
close(f2);
fork();
return 0;
}
```

BANKERSALG

```
#include<stdio.h>
int main (){
int allocated[15][15], max[15][15], need[15][15],
avail[15], tres[15], work[15], flag[15];
int pno, rno, i, j, prc,
count, t, total; count =0;
//clrscr ();
printf ("\n Enter number ofprocess:");scanf ("%d", &pno);
printf ("\n Enter numberofresources:");scanf ("%d", &rno);
for (i = 1; i <= pno; i++){ flag[i] = 0; }
printf ("\n Enter total numbers ofeach resources:");
for (i = 1; i <= rno; i++)
scanf ("%d", &tres[i]);
printf ("\n Enter Max resources foreach process:");
for (i = 1; i <= pno; i++){
printf ("\n for process %d:",i);
for (j = 1; j <= rno; j++)
scanf ("%d", &max[i][j]);
}printf ("\n Enter allocated resources for each
process:"); for (i = 1; i <= pno; i++){
printf ("\n for process %d:",i);
 for (j = 1; j <= rno; j++)
scanf ("%d", &allocated[i][j]);}
printf ("\n available
resources:\n");
for (j = 1; j <= rno; j++){
avail[j] = 0;
total = 0;
for (i = 1; i <= pno; i++){
total += allocated[i][j];}
avail[j] =tres[j] -total;
work[j] =avail[j];
printf (" %d \t", work[j]); }
do{
for (i = 1; i <= pno; i++){
for (j = 1; j <= rno; j++){
need[i][j] = max[i][j] - allocated[i][j]; }
}printf ("\n Allocated matrix Max need");
for (i = 1; i <= pno; i++){ printf ("\n"); for (j = 1; j <= rno; j++){ printf ("%4d", allocated[i][j]); }
printf ("|");
for (j = 1; j <= rno; j++){
printf ("%4d", max[i][j]);}
printf ("|");
for (j = 1; j <= rno; j++){
printf ("%4d", need[i][j]);}}
prc = 0;
for (i = 1; i <= pno; i++){
if (flag[i] == 0){
```

```
prc = i;
for (j = 1; j <= rno; j++){
prc = 0; break; }
if (work[j] < need[i][j]){ } }
if (prc != 0)
break;
}if (prc != 0){
printf ("\n Process %d completed",i);
count++;
printf ("\n Available
matrix:")
for (j = 1; j <= rno; j++){
work[j] +=allocated[prc][j];
allocated[prc][j] = 0;
max[prc][j] = 0;
flag[prc] = 1;
printf (" %d", work[j]);}}}
while (count != pno && prc != 0);
if (count == pno)
printf ("\nThe system is in a safestate!!");
else
printf ("\nThe system is in an unsafestate!!");
return 0;
}
```

PAGING TECHNIQUE

```
#include<stdio.h>
#include<conio.h>
int main()
{
int ms, ps, nop, np, rempages, i, j, x, y, pa, offset;
int s[10], fno[10][20];
printf("\nEnter the memory size -- ");
scanf("%d",&ms);
printf("\nEnter the page size -- ");
scanf("%d",&ps);nop = ms/ps;
printf("\nThe no. of pages available in memory are -- %d ",nop);
printf("\nEnter number of processes -- ");
scanf("%d",&np);
rempages = nop;
for(i=1;i<=np;i++)
{
printf("\nEnter no. of pages required for p[%d]-- ",i);
scanf("%d",&s[i]);
if(s[i] >rempages)
{
printf("\nMemory is Full");
break;
}
rempages = rempages - s[i];
printf("\nEnter pagetable for p[%d] --- ",i);
for(j=0;j<s[i];j++)
scanf("%d",&fno[i][j]);
}
printf("\nEnter Logical Address to find Physical Address ");
printf("\nEnter process no. and page number and offset -- ");
scanf("%d %d %d",&x,&y, &offset);
if(x>np || y>=s[i] || offset>=ps)
printf("\nInvalid Process or Page Number or offset");
else
{ pa=fno[x][y]*ps+offset;
printf("\nThe Physical Address is -- %d",pa);
}
getch();
}
```

PRA/FIFO

```
#include<stdio.h>
int main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
printf("\n ENTER THE NUMBER OF PAGES:\n");
scanf("%d",&n);
printf("\n ENTER THE PAGE NUMBER :\n");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf("\n ENTER THE NUMBER OF FRAMES :");
scanf("%d",&no);
for(i=0;i<no;i++)
frame[i]= -1;
j=0;
printf("\tRef string\t Page Frames\n");
for(i=1;i<=n;i++)
{
printf("%d\t\t\t",a[i]);
avail=0;
for(k=0;k<no;k++)
if(frame[k]==a[i])
avail=1;
if (avail==0)
{
frame[j]=a[i];
j=(j+1)%no;
count++;
for(k=0;k<no;k++)
printf("%d\t",frame[k]);
}
printf("\n");
}
printf("\nPage Fault Is %d",count);
return 0;
}
```

PRA/LRU

```
#include<stdio.h>
main()
{
int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];
printf("Enter no of pages: \n");
scanf("%d",&n);
printf("Enter the reference string: \n");
for(i=0;i<n;i++)
scanf("%d",&p[i]);
printf("Enter no of frames: \n");
scanf("%d",&f);
q[k]=p[k];
printf("\t\n\t %d\n",q[k]);
c++;
k++;
for(i=1;i<n;i++){
c1=0;
for(j=0;j<f;j++){
if(p[i]!=q[j])
c1++;}
if(c1==f){
c++;
if(k<f){
q[k]=p[i];
k++;
for(j=0;j<k;j++)
printf("\t%d",q[j]);
printf("\n");}
else{
for(r=0;r<f;r++){
c2[r]=0;
for(j=i-1;j<n;j--){
if(q[r]!=p[j])
c2[r]++;
else
break;}}
for(r=0;r<f;r++)
b[r]=c2[r];
for(r=0;r<f;r++){
for(j=r;j<f;j++){
if(b[r]<b[j]){
t=b[r];
b[r]=b[j];
b[j]=t;}}}
for(r=0;r<f;r++){
if(c2[r]==b[0])
q[r]=p[i];
printf("\t%d",q[r]);}
printf("\n"); }}}
printf("\nThe no of page faults is %d",c);
}
```

PRA/OPR

```
#include<stdio.h>
int main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
printf("\n ENTER THE NUMBER OF PAGES:\n");
scanf("%d",&n);
printf("\n ENTER THE PAGE NUMBER :\n");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf("\n ENTER THE NUMBER OF FRAMES :");
scanf("%d",&no);
for(i=0;i<no;i++)
frame[i]= -1;
j=0;
printf("\tref string\t page frames\n");
for(i=1;i<=n;i++)
{
printf("%d\t\t",a[i]);
avail=0;
for(k=0;k<no;k++)
if(frame[k]==a[i])
avail=1;
if (avail==0)
{
frame[j]=a[i];
j=(j+1)%no;
count++;
for(k=0;k<no;k++)
printf("%d\t",frame[k]);
}
printf("\n");
}
printf("Page Fault Is %d",count);
return 0;
}
```

DS/FCFS

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int RQ[100],i,n,TotalHeadMoment=0,initial;
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);
for(i=0;i<n;i++)
{
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];
}
printf("Total head moment is %d",TotalHeadMoment);
return 0;
}
```

DS/SSTF

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int RQ[100],i,n,TotalHeadMoment=0,initial,count=0;
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;
}
```

DS/SCAN

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
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);
initial = size-1;
for(i=index-1;i>=0;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);initial =0;
for(i=index;i<n;i++){
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];}}
printf("Total head movement is %d",TotalHeadMoment);
return 0;
}
```

DS/LOOK

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
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];}
for(i=index-1;i>=0;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] ;}
for(i=index;i<n;i++){
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];}}
printf("Total head movement is %d",TotalHeadMoment);
return 0;
}
```

FM/SA

```
#include < stdio.h>
#include<conio.h>
void main()
{
int f[50], i, st, len, j, c, k, count = 0;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
printf("Files Allocated are : \n");
x: count=0;
printf(“Enter starting block and length of files: ”);
scanf("%d%d", &st,&len);
for(k=st;k<(st+len);k++)
if(f[k]==0)
count++;
if(len==count){
for(j=st;j<(st+len);j++)
if(f[j]==0)
{
f[j]=1;
printf("%d\t%d\n",j,f[j]);
}
if(j!=(st+len-1))
printf(” The file is allocated to disk\n");
}
else
printf(” The file is not allocated \n");
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit();
getch();
}
```

FM/IA

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int f[50], index[50],i, n, st, len, j, c, k, ind,count=0;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
x:printf("Enter the index block: ");
scanf("%d",&ind);
if(f[ind]!=1)
{
printf("Enter no of blocks needed and no of files for the index %d on the disk : \n", ind);
scanf("%d",&n);
}
else
{
printf("%d index is already allocated \n",ind);
goto x;
}
y: count=0;
for(i=0;i<n;i++)
{scanf("%d", &index[i]);
if(f[index[i]]==0)
count++;
}
if(count==n)
{
for(j=0;j<n;j++)
f[index[j]]=1;
printf("Allocated\n");
printf("File Indexed\n");
for(k=0;k<n;k++)
printf("%d ------->%d : %d\n",ind,index[k],f[index[k]]);}
else{
printf("File in the index is already allocated \n");
printf("Enter another file indexed");
goto y;}
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit(0);
getch();}
```

FM/LA

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void recursivePart(int pages[]){
int st, len, k, c, j;
printf("Enter the index of the starting block and its length: ");
scanf("%d%d", &st, &len);
k = len;
if (pages[st] == 0){
for (j = st; j < (st + k); j++){
if (pages[j] == 0){
pages[j] = 1;
printf("%d ----- >%d\n", j, pages[j]);
}
else {
printf("The block %d is already allocated \n", j);
k++;}}}
else
printf("The block %d is already allocated \n", st);
printf("Do you want to enter more files? \n");printf("Enter 1 for Yes, Enter 0 for No: ");
scanf("%d", &c);
if (c==1)
recursivePart(pages);
else
exit(0);
return;
}
int main(){
int pages[50], p, a;
for (int i = 0; i < 50; i++)
pages[i] = 0;
printf("Enter the number of blocks already allocated: ");
scanf("%d", &p);
printf("Enter the blocks already allocated: ");
for (int i = 0; i < p; i++){
scanf("%d", &a);
pages[a] = 1;
}
recursivePart(pages);
getch();
return 0;
}
```