SJF

```
//PROGRAM FOR SHORTEST-JOB-FIRST(SJF) "CPU SCHEDULING ALGORITHM" WITHOUT
PRE_EMPTION
#include<stdio.h>
// #include<conio.h>
int main()
{
int at[10], bt[10], ct[10], wt[10], ta[10], tat[10];
//at-ArritvalTime::br-BurstTime::ct-CompletionTime::ta-TemporaryArray
//wt-WaitingTime::tat-TurnAroundTime::tn-CurrentTime(TimeNow)
int n, i, k, pc=0, pointer = 0, tn =0, c;//pc-ProcessesCompleted
char pn[10][10]; //pn-ProcessName
printf("Enter the number of processes: ");
scanf("%d",&n);
printf("Enter <ProcessName> <ArrivalTime> <BurstTime>\n");
for(i=0;i<n;i++)
scanf("%s%d%d",&pn[i],&at[i],&bt[i]);
for(i=0; i<n; i++)
{
ct[i] = -1;
ta[i] = bt[i];
}
while(pc!=n)
{
c = 0;
for(i=0; i<n; i++)
if(ct[i]<0 && at[i]<=tn)
c++;
if(c==0)
tn++;
else
{
pointer = 0;
while(at[pointer]>tn || ct[pointer]>0)
pointer++;
for(k=pointer+1; k<n; k++)
if(at[k]<=tn && ct[k]<0 && bt[pointer]>bt[k])
pointer = k;
if(ct[pointer]<0)
{
tn=tn+bt[pointer];
bt[pointer] = 0;
ct[pointer] = tn;
wt[pointer] = ct[pointer] - ( at[pointer]+ ta[pointer] );
tat[pointer] = ct[pointer] - at[pointer];
pc++;
}
}
}
printf("\nPN\tAT\tBT\tCT\tWT\tTAT\n");
for(i=0;i<n;i++)
printf("%s\t%d\t%d\t%d\t%d\t%d\n",pn[i],at[i],ta[i],ct[i],wt[i],tat[i]);
return 0;}
```

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:");

scanf("%d",&n);

printf("Enter the reference string:");

for(i=0;i<n;i++)

scanf("%d",&p[i]);

printf("Enter no of frames:");

scanf("%d",&f);

q[k]=p[k];

printf("\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);


}
```

Indexed

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct fileTable
{
char name[20];
int nob, blocks[30];
}
ft[30];
void main()
{
int i, j, n;
char s[20];
printf("Enter no of files :");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter file name %d :",i+1);
scanf("%s",ft[i].name);
printf("Enter no ofblocks in file %d :",i+1);
scanf("%d",&ft[i].nob);
printf("Enter the blocks of the file :");
for(j=0;j<ft[i].nob;j++)
scanf("%d",&ft[i].blocks[j]);
}
printf("\nEnter the file name to be searched-- ");
scanf("%s",s); for(i=0;i<n;i++)
if(strcmp(s, ft[i].name)==0) break;
if(i==n)
printf("\nFile Not Found");
else
{
printf("\nFILE NAME NO OF BLOCKS BLOCKS OCCUPIED");
printf("\n %s\t\t%d\t",ft[i].name,ft[i].nob);
for(j=0;j<ft[i].nob;j++)
printf("%d, ",ft[i].blocks[j]);
}
getch();
}
```

Round Robin

```
//PROGRAM FOR ROUND ROBIN "CPU SCHEDULING ALGORITHM" WITH ARRIVAL TIMES

#include<stdio.h>

#include<string.h>

int main(void)

{

//VARIABLE DECLARATION

char pn[20][20], c[20][20]; //PN-PROGRAM NAMES

int n,i,j,k,l, tq, at[20], bt[20], rbt[20], wt[20],tt[20],ct[20]; //bt-BURST TIME ; wt-WAITING TIME; tat-
TURN AROUND TIME

int temp1, temp2, temp3, count=0,twt=0, tn,

tat=0;

printf("Enter <Number_of_Processes & Time_Quantum:\n");

scanf("%d%d", &n, &tq); printf("Enter PN, AT, BT:\n");

//TAKING INPUT VALUES i.e., PROCESS-NAMES, ARRIVAL-TIMES, BURST-TIMES

for(i=0; i<n; i++)

scanf("%s%d%d",pn[i],&at[i],&bt[i]);

for(i=0; i<n; i++)

rbt[i]=bt[i];

//SCHEDULING THE PROCESSES ACCORDING TO SJF

for(i=0;i<n;i++)

{ for(j=i+1; j<n;j++)

{

//SORTING BASED ON ARRIVAL TIMES

if(at[i]>at[j])

{

temp1 = bt[i]; bt[i] = bt[j]; bt[j] = temp1; temp2 = at[i];

at[i] = at[j]; at[j] = temp2; temp3 = rbt[i]; rbt[i] = rbt[j]; rbt[j] = temp3; strcpy(c[i],pn[i]);
strcpy(pn[i],pn[j]); strcpy(pn[j],c[i]);

}

} //END OF J FOR-LOOP }//END OF I FOR-LOOP
tn = at[0]; label:
for(i=0; i<n; i++)
{ if(at[i]>tn) i--; if(rbt[i]>0)
{ if(rbt[i]>tq)
{
   tn += tq; rbt[i] -= tq;
}
else
{
tn += rbt[i]; rbt[i] = 0; ct[i] = tn;
count++;
}
}
}
if(count<n) goto label;
//CALCULATING WAITING TIME & TAT
for(i=0;i<n;i++)
{ wt[i] = ct[i]-at[i]-bt[i]; twt += wt[i];
}
//PRINTING THE VALUES AFTER ALL PREOCESSES COMPLETED
printf("S.N.\tPN\tAT\tBT\tCT\tWT\n");
for(i=0; i<n; i++) printf("%d\t%s\t%d\t%d\t%d\t%d\n",(i+1),pn[i],at[i],bt[i],ct[i],wt[i]);
printf("Total waiting time:%d", twt);
}}//END OF MAIN
```