```
--------------
--------------
CPU SCHEDULING
--------------
--------------


----
FCFS
----

#include<stdio.h>
void main()
{
      int i=0,j=0,b[i],g[20],p[20],w[20],t[20],a[20],n=0,m;
      float avgw=0,avgt=0;
      printf("Enter the number of process : ");
      scanf("%d",&n);
      for(i=0;i<n;i++)
      {
            printf("Process ID : ");
            scanf("%d",&p[i]);

            printf("Burst Time : ");
            scanf("%d",&b[i]);

            printf("Arrival Time: ");
            scanf("%d",&a[i]);
      }

      int temp=0;
      for(i=0;i<n-1;i++)
      {
            for(j=0;j<n-1;j++)
            {
                  if(a[j]>a[j+1])
                  {
temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;

                        temp=b[j];
                        b[j]=b[j+1];
                        b[j+1]=temp;

                        temp=p[j];
                        p[j]=p[j+1];
                        p[j+1]=temp;
                  }
            }
      }

      g[0]=0;
      for(i=0;i<=n;i++)
            g[i+1]=g[i]+b[i];
      for(i=0;i<n;i++)
      {

            t[i]=g[i+1]-a[i];
```

```
            w[i]=t[i]-b[i];
            avgw+=w[i];
            avgt+=t[i];
      }
      avgw=avgw/n;
      avgt=avgt/n;
printf("pid\tarrivalT\tBrustT\tCompletionT\tWaitingtime\tTurnaroundTi\n");
      for(i=0;i<n;i++)
      {
            printf("%d\t%d\t%d\t%d\t\t%d\t\t\t%d\
n",p[i],a[i],b[i],g[i+1],w[i],t[i]);
}
      printf("\nAverage waiting time %f",avgw);
      printf("\nAverage turnarround time %f",avgt);
}


---
SJF
---

#include<stdio.h>
void main()
{
      int i=0,j=0,p[i],b[i],g[20],w[20],t[20],a[20],n=0,m;
      int k=1,min=0,btime=0;
      float avgw=0,avgt=0;
      printf("Enter the number of process : ");
      scanf("%d",&n);
      for(i=0;i<n;i++)
      {
            printf("\nProcess id : ");
            scanf("%d",&p[i]);

            printf("Burst Time : ");
            scanf("%d",&b[i]);
printf("Arrival Time: ");
            scanf("%d",&a[i]);
      }

//sort the jobs based on burst time.
      int temp=0;
      for(i=0;i<n-1;i++)
      {
            for(j=0;j<n-1;j++)
            {
                  if(a[j]>a[j+1])
                  {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;

                        temp=b[j];
                        b[j]=b[j+1];
                        b[j+1]=temp;

                        temp=p[j];
                        p[j]=p[j+1];
                        p[j+1]=temp;
```

```
                  }
            }
      }

      for(i=0;i<n;i++)
      {
            btime=btime+b[i];
            min=b[k];
            for(j=k;j<n;j++)
            {
if(btime >= a[j] && b[j]<min)
                  {
                        temp=a[j];
                        a[j]=a[j-1];
                        a[j-1]=temp;

                        temp=b[j];
                        b[j]=b[j-1];
                        b[j-1]=temp;
temp=p[j];
                        p[j]=p[j-1];
                        p[j-1]=temp;
                  }
            }
            k++;
      }
      g[0]=a[0];
      for(i=0;i<n;i++)
      {
            g[i+1]=g[i]+b[i];
            if(g[i]<a[i])
                  g[i]=a[i];
      }
      for(i=0;i<n;i++)
      {


            t[i]=g[i+1]-a[i];
            w[i]=t[i]-b[i];
            avgw+=w[i];
            avgt+=t[i];
      }
      avgw=avgw/n;
      avgt=avgt/n;
      printf("pid\tBrustTime\tGantChart\tWaiting time\t\tTurnarround Time\n");
      for(i=0;i<n;i++)
      {
            printf(" %d\t %d\t\t%d-%d\t\t%d\t\t\t%d\
n",p[i],b[i],g[i],g[i+1],w[i],t[i]);
      }
      printf("\nAverage waiting time %f",avgw);
      printf("\nAverage turnarround time %f\n",avgt);

}



----
PRIO
```

```
#include<stdio.h>
int main()
{
      int burst_time[20], process[20], waiting_time[20], turnaround_time[20],
priority[20];
      int i, j, limit, sum = 0, position, temp;
      float average_wait_time, average_turnaround_time;
      printf("Enter Total Number of Processes:\t");
      scanf("%d", &limit);
      printf("\nEnter Burst Time and Priority For %d Processes\n", limit);
      for(i = 0; i < limit; i++)
      {
            printf("\nProcess[%d]\n", i + 1);
            printf("Process Burst Time:\t");
            scanf("%d", &burst_time[i]);
            printf("Process Priority:\t");
            scanf("%d", &priority[i]);
            process[i] = i + 1;
      }
      for(i = 0; i < limit; i++)
      {
            position = i;
            for(j = i + 1; j < limit; j++)
            {
                  if(priority[j] < priority[position])
                  {
                        position = j;
                  }
            }
            temp = priority[i];
            priority[i] = priority[position];
            priority[position] = temp;
            temp = burst_time[i];
            burst_time[i] = burst_time[position];
            burst_time[position] = temp;
            temp = process[i];
            process[i] = process[position];
            process[position] = temp;
      }
      waiting_time[0] = 0;
      for(i = 1; i < limit; i++)
      {
            waiting_time[i] = 0;
            for(j = 0; j < i; j++)
            {
                  waiting_time[i] = waiting_time[i] + burst_time[j];
            }
            sum = sum + waiting_time[i];
      }
      average_wait_time = sum / limit;
      sum = 0;
      printf("\nProcess ID\t\tBurst Time\t Waiting Time\t Turnaround Time\n");
      for(i = 0; i < limit; i++)
      {
            turnaround_time[i] = burst_time[i] + waiting_time[i];
            sum = sum + turnaround_time[i];
```

```
            printf("\nProcess[%d]\t\t%d\t\t %d\t\t %d\n", process[i],
burst_time[i], waiting_time[i], turnaround_time[i]);
      }
      average_turnaround_time = sum / limit;
      printf("\nAverage Waiting Time:\t%f", average_wait_time);
      printf("\nAverage Turnaround Time:\t%f\n", average_turnaround_time);
      return 0;
}
```

ROUND ROBIN

```
#include<stdio.h>
 int main()
{
      int i, limit, total = 0, x, counter = 0, time_quantum;
      int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10],
temp[10];
      float average_wait_time, average_turnaround_time;
      printf("\nEnter Total Number of Processes:\t");
      scanf("%d", &limit);
      x = limit;
      for(i = 0; i < limit; i++)
      {
            printf("\nEnter Details of Process[%d]\n", i + 1);
            printf("Arrival Time:\t");
            scanf("%d", &arrival_time[i]);
            printf("Burst Time:\t");
            scanf("%d", &burst_time[i]);
            temp[i] = burst_time[i];
      }
      printf("\nEnter Time Quantum:\t");
      scanf("%d", &time_quantum);
      printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n");
      for(total = 0, i = 0; x != 0;)
      {
            if(temp[i] <= time_quantum && temp[i] > 0)
            {
                  total = total + temp[i];
                  temp[i] = 0;
                  counter = 1;
            }
            else if(temp[i] > 0)
            {
                  temp[i] = temp[i] - time_quantum;
                  total = total + time_quantum;
            }
            if(temp[i] == 0 && counter == 1)
            {
                  x--;
                  printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1,
burst_time[i], total - arrival_time[i], total - arrival_time[i] - burst_time[i]);
                  wait_time = wait_time + total - arrival_time[i] - burst_time[i];
                  turnaround_time = turnaround_time + total - arrival_time[i];
                  counter = 0;
```

```
            }
            if(i == limit - 1)
            {
                  i = 0;
            }
            else if(arrival_time[i + 1] <= total)
            {
                  i++;
            }
            else
            {
                  i = 0;
            }
      }
      average_wait_time = wait_time * 1.0 / limit;
      average_turnaround_time = turnaround_time * 1.0 / limit;
      printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
      printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
      return 0;
}
```

BANKER

```
#include<stdio.h>
struct pro{
      int all[10],max[10],need[10];
      int flag;
};
int i,j,pno,r,nr,id,k=0,safe=0,exec,count=0,wait=0,max_err=0;
struct pro p[10];
int aval[10],seq[10];
void safeState()
{
      while(count!=pno){
            safe = 0;
            for(i=0;i<pno;i++){
                  if(p[i].flag){
                        exec = r;
                        for(j=0;j<r;j++)
                        {
                              if(p[i].need[j]>aval[j]){
                                    exec =0;
                              }
                        }
                        if(exec == r){
                              for(j=0;j<r;j++){
                                    aval[j]+=p[i].all[j];
                              }
                              p[i].flag = 0;
                              seq[k++] = i;
                              safe = 1;
                              count++;

                        }
```

```
                  }
            }
            if(!safe)
            {
                  printf("System is in Unsafe State\n");
                  break;
            }
      }
      if(safe){
            printf("\n\nSystem is in safestate \n");
            printf("Safe State Sequence \n");
            for(i=0;i<k;i++)
                  printf("P[%d]   ",seq[i]);
            printf("\n\n");
      }
}
void reqRes(){
      printf("\nRequest for new Resourses");
      printf("\nProcess id ? ");
      scanf("%d",&id);
      printf("Enter new Request details ");
      for(i=0;i<r;i++){

            scanf("%d",&nr);
            if( nr <= p[id].need[i])
            {
                  if( nr <= aval[i]){
                        aval[i] -= nr;
                        p[id].all[i] += nr;
                        p[id].need[i] -= nr;
                  }
                  else
                        wait = 1;
            }
            else
                  max_err = 1;
      }
      if(!max_err && !wait)
            safeState();
      else if(max_err){
            printf("\nProcess has exceeded its maximum usage \n");
      }
      else{
            printf("\nProcess need to wait\n");
      }


}
void main()
{

      printf("Enter no of process ");
      scanf("%d",&pno);

      printf("Enter no. of resourses ");
      scanf("%d",&r);

      printf("Enter Available Resourse of each type ");
      for(i=0;i<r;i++){
```

```
		scanf("%d",&aval[i]);
	}

	printf("\n\n---Resourse Details---");
	for(i=0;i<pno;i++){

		printf("\nResourses for process %d\n",i);
		printf("\nAllocation Matrix\n");
		for(j=0;j<r;j++){
			scanf("%d",&p[i].all[j]);
		}
		printf("Maximum Resourse Request \n");
		for(j=0;j<r;j++){
			scanf("%d",&p[i].max[j]);
		}
		p[i].flag = 1;

	}
	// Calcualting need
	for(i=0;i<pno;i++){
		for(j=0;j<r;j++){
			p[i].need[j] = p[i].max[j] - p[i].all[j];
		}
	}

	//Print Current Details
	printf("\nProcess Details\n");
	printf("Pid\t\tAllocattion\t\tMax\t\tNeed\n");
	for(i=0;i<pno;i++)
	{
		printf("%d\t\t",i);
		for(j=0;j<r;j++){
			printf("%d  ",p[i].all[j]);
		}
		printf("\t\t");
		for(j=0;j<r;j++){
			printf("%d  ",p[i].max[j]);
		}
		printf("\t\t");
		for(j=0;j<r;j++){
			printf("%d  ",p[i].need[j]);
		}
		printf("\n");

	}

	//Determine Current State in Safe State
	safeState();
	int ch=1;
	do{
		printf("Request new resourse ?[0/1] :");
		scanf("%d",&ch);
		if(ch)
			reqRes();
	}while(ch!=0);

	//end:printf("\n");

}
```


DISK SCHEDULING

FCFS

```
#include<stdio.h>
void main(){

        int ioq[20],i,n,ihead,tot;
        float seek=0,avgs;

        printf("Enter the number of requests\t:");
        scanf("%d",&n);
        printf("Enter the initial head position\t:");
        scanf("%d",&ihead);
        ioq[0] = ihead;
        ioq[n+1] =0;

        printf("Enter the I/O queue requests \n");
        for(i=1;i<=n;i++){
                scanf("%d",&ioq[i]);
        }
        ioq[n+1] =ioq[n];// to set the last seek zero

        printf("\nOrder of request served\n");
        for(i=0;i<=n;i++){

                tot = ioq[i+1] - ioq[i];
                if(tot < 0)
                        tot = tot * -1;
                seek += tot;
               // printf("%d\t%d\n",ioq[i],tot);// to display each seek
                printf("%d --> ",ioq[i]);

        }
avgs = seek/(n);

        printf("\nTotal Seek time\t\t: %.2f",seek);
        printf("\nAverage seek time\t: %.2f\n\n",avgs);
}
```

SCAN

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

```
void main()
{
        int ioq[20],i,n,j,ihead,temp,scan,tot;
        float seek=0,avgs;

        printf("Enter the number of requests\t:");
        scanf("%d",&n);
        printf("Enter the initial head position\t:");
        scanf("%d",&ihead);
        ioq[0] = ihead;
        ioq[1] = 0;
        n += 2;
        printf("Enter the I/O queue requests \n");
        for(i=2;i<n;i++){
                scanf("%d",&ioq[i]);
        }

        for(i=0;i<n-1;i++){
                for(j=0;j<n-1;j++)
                {

                        if(ioq[j] > ioq[j+1]){

                                temp = ioq[j];
                                ioq[j] = ioq[j+1];
                                ioq[j+1] = temp;

                        }

                }
        }
        ioq[n]=ioq[n-1];
        for(i=0;i<n;i++){

                if(ihead == ioq[i]){

                        scan = i;
                        break;
                }

        }

        printf("\nOrder of request served\n\n");
        tot = 0;
        for(i=scan;i>=0;i--){
                 //rai  tot = ioq[i+1] - ioq[i];
                 tot = ioq[i] – ioq[i-1]; // me
                 if(i==0) // me
                        tot=ioq[i]-ioq[scan+1];//me
                if(tot < 0)
                        tot = tot * -1;
                //seek += tot;
                printf("%d\t%d\n",ioq[i],tot);
        }


        for(i=scan+1;i<n;i++){
                tot = ioq[i+1] - ioq[i];
                if(tot < 0)
```

```
                        tot = tot * -1;
                //seek += tot;
                printf("%d\t%d\n",ioq[i],tot);
        }
        seek = ihead + ioq[n-1];

        avgs = seek/(n-2);

        printf("\n\nTotal Seek time\t\t: %.2f",seek);
        printf("\nAverage seek time\t: %.2f\n\n",avgs);

}
```

C-SCAN

```
#include<stdio.h>
void main()
{
        int ioq[20],i,n,j,ihead,itail,temp,scan,tot=0;
        float seek=0,avgs;

        printf("Enter the number of requests\t: ");
        scanf("%d",&n);
        ioq[0] = 0;
        printf("Enter the initial head position\t: ");
        scanf("%d",&ihead);
        ioq[1] = ihead;
        printf("Enter the maximum track limit\t: ");
        scanf("%d",&itail);
        ioq[2] = itail;
        n += 3;

        printf("Enter the I/O queue requests \n");
        for(i=3;i<n;i++){
                scanf("%d",&ioq[i]);
        }


        for(i=0;i<n-1;i++){
                for(j=0;j<n-1;j++)
                {

                        if(ioq[j] > ioq[j+1]){

                                temp = ioq[j];
                                ioq[j] = ioq[j+1];
                                ioq[j+1] = temp;

                        }

                }
        }

        for(i=0;i<n+1;i++){
```

```
                if(ihead == ioq[i]){

                        scan = i;
                        break;

                }

        }

        i = scan;
        temp = n;

        printf("\nOrder of request served\n");
        printf("\n");

        while(i != temp){

                if(i < temp-1){
                        tot = ioq[i+1] - ioq[i];

                        if(tot < 0)
                                tot = tot * -1;
                        seek += tot;
                }
                printf("%d --> ",ioq[i]);
          // printf("%d\t%d\n",ioq[i],tot);
                i++;

                if(i == n){
                        i = 0;
                        temp = scan;
                        seek += itail;

                }

        }

         avgs = seek/(n-3);

        printf("\n\nTotal Seek time\t\t: %.2f",seek);
        printf("\nAverage seek time\t: %.2f\n\n",avgs);
 }
```

---
pass1

```
#include<stdio.h>
#include<string.h>
void main()
{
FILE *f1,*f2,*f3,*f4;
char s[100],lab[30],opcode[30],opa[30],opcode1[30],opa1[30];
int locctr,x=0;
f1=fopen("input.txt","r");
```

```
f2=fopen("opcode.txt","r");
f3=fopen("out1.txt","w");
f4=fopen("sym1.txt","w");
while(fscanf(f1,"%s%s%s",lab,opcode,opa)!=EOF)
{
if(strcmp(lab,"**")==0)
{
if(strcmp(opcode,"START")==0)
{
fprintf(f3,"%s %s %s",lab,opcode,opa);
locctr=(atoi(opa));
}
Else
{
rewind(f2);
x=0;
while(fscanf(f2,"%s%s",opcode1,opa1)!=EOF)
{
if(strcmp(opcode,opcode1)==0)
{
x=1;
}}
if(x==1)
{
fprintf(f3,"\n %d %s %s %s",locctr,lab,opcode,opa);
locctr=locctr+3;
}}}
else
{
if(strcmp(opcode,"RESW")==0)
{
fprintf(f3,"\n %d %s %s %s",locctr,lab,opcode,opa);
fprintf(f4,"\n %d %s",locctr,lab);
locctr=locctr+(3*(atoi(opa)));
}
else if(strcmp(opcode,"WORD")==0)
{
fprintf(f3,"\n %d %s %s %s",locctr,lab,opcode,opa);
fprintf(f4,"\n %d %s",locctr,lab);
locctr=locctr+3;
}
else if(strcmp(opcode,"BYTE")==0)
{
fprintf(f3,"\n %d %s %s %s",locctr,lab,opcode,opa);
fprintf(f4,"\n %d %s",locctr,lab);
locctr=locctr+1;
}
else if(strcmp(opcode,"RESB")==0)
{
fprintf(f3,"\n %d %s %s %s",locctr,lab,opcode,opa);
fprintf(f4,"\n %d %s",locctr,lab);
locctr=locctr+1;
}
else
{
fprintf(f3,"\n %d %s %s %s",locctr,lab,opcode,opa);
fprintf(f4,"\n %d %s",locctr,lab);
locctr=locctr+(atoi(opa));
}}}}
```

---
pass2

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
char opcode[20],operand[20],symbol[20],label[20],code[20],mnemonic[25],
character,add[20],objectcode[20];
int flag,flag1,locctr,location,loc;
FILE *fp1,*fp2,*fp3,*fp4;
fp1=fopen("out3.txt","r"); fp2=fopen("twoout.txt","w");
fp3=fopen("opcode.txt","r"); fp4=fopen("sym1.txt","r");
fscanf(fp1,"%s%s%s",label,opcode,operand);
if(strcmp(opcode,"START")==0)
{
fprintf(fp2,"%s\t%s\t%s\n",label,opcode,operand);
fscanf(fp1,"%d%s%s%s",&locctr,label,opcode,operand);
}
while(strcmp(opcode,"END")!=0)
{
flag=0;
fscanf(fp3,"%s%s",code,mnemonic);
while(strcmp(code,"END")!=0)
{
if((strcmp(opcode,code)==0) && (strcmp(mnemonic,"*"))!=0)
{
flag=1;
break;
}
fscanf(fp3,"%s%s",code,mnemonic);
}
if(flag==1)
{
flag1=0; rewind(fp4);
while(!feof(fp4))
{
fscanf(fp4,"%s%d",symbol,&loc);
if(strcmp(symbol,operand)==0)
{
flag1=1; break;
}}
if(flag1==1)
{
sprintf(add,"%d",loc);
strcpy(objectcode,strcat(mnemonic,add));
}}
else if(strcmp(opcode,"BYTE")==0 || strcmp(opcode,"WORD")==0)
{
if((operand[0]=='C') || (operand[0]=='X'))
{
character=operand[2];
sprintf(add,"%d",character);
```

```
strcpy(objectcode,add);
}
Else
{
strcpy(objectcode,add);
}}
else
strcpy(objectcode,"\0");
fprintf(fp2,"%s\t%s\t%s\t%d\t%s\n",label,opcode,operand,locctr,objectcode);
fscanf(fp1,"%d%s%s%s",&locctr,label,opcode,operand);
}
fprintf(fp2,"%s\t%s\t%s\t%d\n",label,opcode,operand,locctr);
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
}
```