**Operating Systems Algorithms**

**Bankers Algorithm**

**#include <iostream>**

**using namespace std;**

**int bankersAlgorithm(int available[1][50],int need[50][50],int nump,int numr,int a[50],int allocated[50][50]){**

**int ctr=0,m=0;**

**int work[1][50];**

**bool finish[50];**

**bool flag=false;**

**for(int j=0;j<numr;j++){**

**work[0][j]=available[0][j];**

**}**

**for(int i=0;i<nump;i++){**

**finish[i]=false;**

**}**

**for(int k=0;k<nump;k++){**

**for(int i=0;i<nump;i++){**

**if(finish[i]==0){**

**flag=false;**

**}**

**for(int j=0;j<numr;j++){**

**if(need[i][j]>work[0][j]){**

**flag=true;**

**}**

**}**

**if(flag==0 && finish[i]==0){**

**for(int j=0;j<numr;j++){**

**work[0][j]+=allocated[i][j];**

**}**

**finish[i]=true;**

**ctr++;**

**a[m]=i;**

**m++;**

**}**

**}**

**}**

**//cout<<ctr<<endl;**

**if(ctr==nump){**

**return 1;**

**}**

**return 0;**

**}**

**int main(){**

**int nump,numr,maximum[50][50],allocated[50][50],need[50][50],available[1][50],x[50],y[50];**

**char ch;**

**cout<<"Enter the number of proceses: ";**

**cin>>nump;**

**cout<<"Enter the number of resources that each process require: ";**

**cin>>numr;**

**cout<<"Enter the entries of maximum requirement matrix: ";**

**for(int i=0;i<nump;i++){**

**for(int j=0;j<numr;j++){**

**cin>>maximum[i][j];**

**}**

**}**

**cout<<"Enter the entries of allocation matrix: ";**

**for(int i=0;i<nump;i++){**

**for(int j=0;j<numr;j++){**

**cin>>allocated[i][j];**

**}**

**}**

**for(int i=0;i<nump;i++){**

**for(int j=0;j<numr;j++){**

**need[i][j]=maximum[i][j]-allocated[i][j];**

**}**

**}**

**cout<<"Enter the available resources: "<<endl;**

**for(int j=0;j<numr;j++){**

**cin>>available[0][j];**

**}**

**cout<<"Allocation Matrix:"<<endl;**

**for(int i=0;i<nump;i++){**

**for(int j=0;j<numr;j++){**

**cout<<allocated[i][j]<<" ";**

**}**

**cout<<endl;**

**}**

**cout<<"Maximum Requirement Matrix:"<<endl;**

**for(int i=0;i<nump;i++){**

**for(int j=0;j<numr;j++){**

**cout<<maximum[i][j]<<" ";**

**}**

**cout<<endl;**

**}**

**cout<<"Need Matrix:"<<endl;**

**for(int i=0;i<nump;i++){**

**for(int j=0;j<numr;j++){**

**cout<<need[i][j]<<" ";**

**}**

**cout<<endl;**

**}**

**int ctr=bankersAlgorithm(available,need,nump,numr,x,allocated);**

**if(ctr==1){**

**cout<<"The sequence is in the safe state."<<endl;**

**for(int i=0;i<nump;i++){**

**cout<<"P"<<x[i]+1<<" ";**

**}**

**cout<<endl;**

**}**

**else{**

**cout<<"The safe sequence is not formed. The system is in deadlock state."<<endl;**

**exit(0);**

**}**

**cout<<"Do you want to add new request (yes-y/no-n)? ";**

**cin>>ch;**

**if(ch=='y'){**

**int request[1][50],a;**

**cout<<"Enter the process number: ";**

**cin>>a;**

**cout<<"Enter the process request: ";**

**for(int j=0;j<numr;j++){**

**cin>>request[0][j];**

**}**

**bool flag1=false;**

**for(int j=0;j<numr;j++){**

**if(request[0][j]>need[a-1][j]){**

**flag1=true;**

**}**

**}**

**if(flag1){**

**cout<<"The process has exceeded its maximum claim."<<endl;**

**exit(0);**

**}**

**bool flag2=false;**

**for(int j=0;j<numr;j++){**

**if(request[0][j]>available[0][j]){**

**flag2=true;**

**}**

**}**

**if(flag2){**

**cout<<"The process P"<<a<<"must wait, since the resources are not available."<<endl;**

**exit(0);**

**}**

**for(int j=0;j<numr;j++){**

**available[0][j]-=request[0][j];**

**allocated[a-1][j]+=request[0][j];**

**need[a-1][j]-=request[0][j];**

**}**

**int ctr1=bankersAlgorithm(available,need,nump,numr,y,allocated);**

**if(ctr1==1){**

**cout<<"The sequence is in the safe state."<<endl;**

**for(int i=0;i<nump;i++){**

**cout<<"P"<<y[i]+1<<" ";**

**}**

**cout<<endl;**

**}**

**else{**

**cout<<"The safe sequence is not formed. The system is in deadlock state."<<endl;**

**}**

**}**

**return 0;**

**}**

**CPU Scheduling Algorithms**

***First Come First Served***

**#include <iostream>**

**#include <iomanip>**

**#include <cmath>**

**using namespace std;**

**void resort(float aTime[50],float bTime[50],float tTime[50],float wTime[50],int info[50],int n)**

**{**

**for(int i=0;i<n-1;i++)**

**{**

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

**{**

**if(info[j]>info[j+1])**

**{**

**int temp1=info[j];**

**info[j]=info[j+1];**

**info[j+1]=temp1;**

**float temp2=tTime[j];**

**tTime[j]=tTime[j+1];**

**tTime[j+1]=temp2;**

**float temp3=wTime[j];**

**wTime[j]=wTime[j+1];**

**wTime[j+1]=temp3;**

**float temp4=aTime[j];**

**aTime[j]=aTime[j+1];**

**aTime[j+1]=temp4;**

**float temp5=bTime[j];**

**bTime[j]=bTime[j+1];**

**bTime[j+1]=temp5;**

**}**

**}**

**}**

**}**

**void bandwTime(float aTime[50],float bTime[50],float wTime[50],float tTime[50],float gcTime[2][102],int n,int m){**

**int i=0,k=0;**

**float extra[50];**

**while(i<n && k<m){**

**if(gcTime[1][k]==0){**

**extra[i]=gcTime[0][k+1];**

**i++;**

**k+=2;**

**}**

**else{**

**extra[i]=gcTime[0][k];**

**i++;**

**k++;**

**}**

**}**

**for(int i=0;i<n;i++){**

**tTime[i]=abs(extra[i]-aTime[i]);**

**wTime[i]=abs(tTime[i]-bTime[i]);**

**}**

**}**

**int gangChart(float gcTime[2][102],float bTime[50],float aTime[50],int PID[50],int n)**

**{**

**int i=0,k=0;**

**if(aTime[0]!=0){**

**gcTime[0][0]=aTime[0];**

**gcTime[1][0]=0;**

**gcTime[0][1]=gcTime[0][0]+bTime[0];**

**gcTime[1][1]=PID[0];**

**i++;**

**k+=2;**

**}**

**else{**

**gcTime[0][0]=bTime[0];**

**gcTime[1][0]=PID[0];**

**i++;**

**k++;**

**}**

**while(i<n){**

**if(gcTime[0][k-1]<aTime[i]){**

**float IT=aTime[i]-gcTime[0][k-1];**

**gcTime[0][k]=gcTime[0][k-1]+IT;**

**gcTime[1][k]=0;**

**k++;**

**}**

**gcTime[0][k]=gcTime[0][k-1]+bTime[i];**

**gcTime[1][k]=PID[i];**

**i++;**

**k++;**

**}**

**return k;**

**}**

**void sorting(float aTime[50],float bTime[50],int PID[50],int n)**

**{**

**for(int i=0;i<n-1;i++)**

**{**

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

**{**

**if(aTime[j]>aTime[j+1])**

**{**

**float temp1=aTime[j];**

**aTime[j]=aTime[j+1];**

**aTime[j+1]=temp1;**

**float temp2=bTime[j];**

**bTime[j]=bTime[j+1];**

**bTime[j+1]=temp2;**

**int temp3=PID[j];**

**PID[j]=PID[j+1];**

**PID[j+1]=temp3;**

**}**

**}**

**}**

**}**

**int main(){**

**float aTime[50],bTime[50],wTime[50],tTime[50],gcTime[2][102];**

**int n,PID[50];**

**cout<<"Enter the no of processes: ";**

**cin>>n;**

**cout<<"Enter the burst time: ";**

**for(int i=0;i<n;i++){**

**cin>>bTime[i];**

**}**

**cout<<"Enter the arrival time: ";**

**for(int i=0;i<n;i++){**

**cin>>aTime[i];**

**}**

**for(int i=0;i<n;i++){**

**PID[i]=i+1;**

**}**

**sorting(aTime,bTime,PID,n);**

**int m=gangChart(gcTime,bTime,aTime,PID,n);**

**bandwTime(aTime,bTime,wTime,tTime,gcTime,n,m);**

**float sum1=0,sum2=0,avgTT,avgWT;**

**for(int i=0;i<n;i++){**

**sum1+=tTime[i];**

**sum2+=wTime[i];**

**}**

**avgTT=sum1/n;**

**avgWT=sum2/n;**

**int info[50];**

**int i=0,k=0;**

**while(i<n){**

**if(gcTime[1][k]==0){**

**k++;**

**}**

**info[i]=gcTime[1][k];**

**i++;**

**k++;**

**}**

**resort(aTime,bTime,tTime,wTime,info,n);**

**cout<<"Process"<<" "<<"Arrival Time"<<" "<<"Burst Time"<<" "<<"Turnaround Time"<<" "<<"Waiting Time"<<endl;**

**for(int i=0;i<n;i++){**

**cout<<setw(4)<<"P"<<i+1<<" "<<aTime[i]<<" "<<bTime[i]<<" "<<tTime[i]<<" "<<wTime[i]<<endl;**

**}**

**cout<<"The average turnaround time is = "<<avgTT<<endl;**

**cout<<"The average waiting time is = "<<avgWT<<endl;**

**return 0;**

**}**

***Shortest Job First***

**#include <iostream>**

**#include <iomanip>**

**#include <cmath>**

**using namespace std;**

**bool idleTimeChecker(float gcmax,float aTime[50],int index,int n){**

**for(int j=index;j<n;j++){**

**if(gcmax<aTime[j]){**

**return true;**

**}**

**}**

**return false;**

**}**

**void resort(float tTime[50],float wTime[50],int info[50],int n)**

**{**

**for(int i=0;i<n-1;i++)**

**{**

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

**{**

**if(info[j]>info[j+1])**

**{**

**int temp1=info[j];**

**info[j]=info[j+1];**

**info[j+1]=temp1;**

**float temp2=tTime[j];**

**tTime[j]=tTime[j+1];**

**tTime[j+1]=temp2;**

**float temp3=wTime[j];**

**wTime[j]=wTime[j+1];**

**wTime[j+1]=temp3;**

**}**

**}**

**}**

**}**

**void bandwTime(float aTime[50],float bTime[50],float wTime[50],float tTime[50],float gcTime[2][102],int n,int m){**

**int i=0,k=0;**

**float extra[50];**

**int id[50];**

**while(i<n && k<m){**

**if(gcTime[1][k]==0){**

**extra[i]=gcTime[0][k+1];**

**id[i]=(int)gcTime[1][k+1];**

**i++;**

**k+=2;**

**}**

**else{**

**extra[i]=gcTime[0][k];**

**id[i]=(int)gcTime[1][k];**

**i++;**

**k++;**

**}**

**}**

**for(int i=0;i<n;i++){**

**int indi=id[i]-1;**

**tTime[i]=abs(extra[i]-aTime[indi]);**

**wTime[i]=abs(tTime[i]-bTime[indi]);**

**}**

**}**

**int gangChart(float gcTime[2][102],float bTime[50],float aTime[50],int PID[50],int n)**

**{**

**int i=0,k=0;**

**if(aTime[0]!=0){**

**gcTime[0][0]=aTime[0];**

**gcTime[1][0]=0;**

**gcTime[0][1]=gcTime[0][0]+bTime[0];**

**gcTime[1][1]=PID[0];**

**i++;**

**k+=2;**

**}**

**else{**

**gcTime[0][0]=bTime[0];**

**gcTime[1][0]=PID[0];**

**i++;**

**k++;**

**}**

**while(i<n){**

**float gcmax=gcTime[0][k-1];**

**int index=i;**

**bool ic=idleTimeChecker(gcmax,aTime,index,n);**

**if(ic){**

**float IT=aTime[i]-gcTime[0][k-1];**

**gcTime[0][k]=gcTime[0][k-1]+IT;**

**gcTime[1][k]=0;**

**k++;**

**float gcmax1=gcTime[0][k-1];**

**int index1=i;**

**bool ic1=idleTimeChecker(gcmax1,aTime,index1,n);**

**if(ic1){**

**gcTime[0][k]=gcTime[0][k-1]+bTime[i];**

**gcTime[1][k]=PID[i];**

**i++;**

**k++;**

**}**

**else{**

**int j=0;**

**float b[50];**

**int e[50];**

**for(int l=i;l<n;l++){**

**b[j]=bTime[l];**

**e[j]=PID[l];**

**j++;**

**}**

**i+=j;**

**for(int p=0;p<j-1;p++)**

**{**

**for(int q=0;q<j-p-1;q++)**

**{**

**if(b[q]>b[q+1])**

**{**

**float temp=b[q];**

**b[q]=b[q+1];**

**b[q+1]=temp;**

**int temp1=e[q];**

**e[q]=e[q+1];**

**e[q+1]=temp1;**

**}**

**}**

**}**

**for(int p=0;p<j;p++){**

**gcTime[0][k]=gcTime[0][k-1]+b[p];**

**gcTime[1][k]=e[p];**

**k++;**

**}**

**}**

**}**

**else{**

**int j=0;**

**float b[50];**

**int e[50];**

**for(int l=i;l<n;l++){**

**b[j]=bTime[l];**

**e[j]=PID[l];**

**j++;**

**}**

**i+=j;**

**for(int p=0;p<j-1;p++)**

**{**

**for(int q=0;q<j-p-1;q++)**

**{**

**if(b[q]>b[q+1])**

**{**

**float temp=b[q];**

**b[q]=b[q+1];**

**b[q+1]=temp;**

**int temp1=e[q];**

**e[q]=e[q+1];**

**e[q+1]=temp1;**

**}**

**}**

**}**

**for(int p=0;p<j;p++){**

**gcTime[0][k]=gcTime[0][k-1]+b[p];**

**gcTime[1][k]=e[p];**

**k++;**

**}**

**}**

**}**

**return k;**

**}**

**void sorting(float aTime[50],float bTime[50],int PID[50],int n)**

**{**

**for(int i=0;i<n-1;i++)**

**{**

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

**{**

**if(aTime[j]>aTime[j+1])**

**{**

**float temp1=aTime[j];**

**aTime[j]=aTime[j+1];**

**aTime[j+1]=temp1;**

**float temp2=bTime[j];**

**bTime[j]=bTime[j+1];**

**bTime[j+1]=temp2;**

**int temp3=PID[j];**

**PID[j]=PID[j+1];**

**PID[j+1]=temp3;**

**}**

**}**

**}**

**}**

**int main(){**

**float aTime[50],bTime[50],wTime[50],tTime[50],gcTime[2][102];**

**int n,PID[50];**

**float ats[50],bts[50];**

**cout<<"Enter the no of processes: ";**

**cin>>n;**

**cout<<"Enter the burst time: ";**

**for(int i=0;i<n;i++){**

**cin>>bTime[i];**

**bts[i]=bTime[i];**

**}**

**cout<<"Enter the arrival time: ";**

**for(int i=0;i<n;i++){**

**cin>>aTime[i];**

**ats[i]=aTime[i];**

**}**

**for(int i=0;i<n;i++){**

**PID[i]=i+1;**

**}**

**sorting(ats,bts,PID,n);**

**int m=gangChart(gcTime,bts,ats,PID,n);**

**bandwTime(aTime,bTime,wTime,tTime,gcTime,n,m);**

**float sum1=0,sum2=0,avgTT,avgWT;**

**for(int i=0;i<n;i++){**

**sum1+=tTime[i];**

**sum2+=wTime[i];**

**}**

**avgTT=sum1/n;**

**avgWT=sum2/n;**

**int info[50];**

**int i=0,k=0;**

**while(i<n){**

**if(gcTime[1][k]==0){**

**k++;**

**}**

**info[i]=gcTime[1][k];**

**i++;**

**k++;**

**}**

**resort(tTime,wTime,info,n);**

**cout<<"Process"<<" "<<"Arrival Time"<<" "<<"Burst Time"<<" "<<"Turnaround Time"<<" "<<"Waiting Time"<<endl;**

**for(int i=0;i<n;i++){**

**cout<<setw(4)<<"P"<<i+1<<" "<<aTime[i]<<" "<<bTime[i]<<" "<<tTime[i]<<" "<<wTime[i]<<endl;**

**}**

**cout<<"The average turnaround time is = "<<avgTT<<endl;**

**cout<<"The average waiting time is = "<<avgWT<<endl;**

**return 0;**

**}**

***Shortest Remaining Time First***

**#include<iostream>**

**#include<iomanip>**

**using namespace std;**

**struct process{**

**int PID;**

**int bTime;**

**int aTime;**

**int cTime;**

**int sTime;**

**int tTime;**

**int wTime;**

**};**

**int main(){**

**process p[50];**

**int n;**

**cout<<"Enter the no of process: ";**

**cin>>n;**

**cout<<"Enter the arrival time of the processes: ";**

**for(int i=0;i<n;i++){**

**cin>>p[i].aTime;**

**}**

**cout<<"Enter the burst time of the process: ";**

**for(int i=0;i<n;i++){**

**cin>>p[i].bTime;**

**}**

**for(int i=0;i<n;i++){**

**p[i].PID=i+1;**

**}**

**int isCompleted[50];**

**for(int i=0;i<50;i++){**

**isCompleted[i]=0;**

**}**

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

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

**if(p[j].aTime>p[j+1].aTime){**

**int temp1=p[j].aTime;**

**p[j].aTime=p[j+1].aTime;**

**p[j+1].aTime=temp1;**

**int temp2=p[j].bTime;**

**p[j].bTime=p[j+1].bTime;**

**p[j+1].bTime=temp2;**

**int temp3=p[j].PID;**

**p[j].PID=p[j+1].PID;**

**p[j+1].PID=temp3;**

**}**

**}**

**}**

**int currTime=0;**

**int compTime=0;**

**int prevT=0;**

**int totalTT=0;**

**int totalWT=0;**

**int br[50];**

**int totalIT=0;**

**for(int i=0;i<n;i++){**

**br[i]=p[i].bTime;**

**}**

**while(compTime!=n){**

**int idle=-1;**

**int mini=10000000;**

**//Checking for process for shortest burst time that are in ready queue at that time**

**for(int i=0;i<n;i++){**

**if(p[i].aTime<=currTime && isCompleted[i]==0){**

**if(br[i]<mini){**

**mini=br[i];**

**idle=i;**

**}**

**if(br[i]==mini){**

**if(p[i].aTime<p[idle].aTime){**

**mini=br[i];**

**idle=i;**

**}**

**}**

**}**

**}**

**//To check whether the process is found or not**

**if(idle!=-1){**

**//To check whether the process arrives for the first time or not**

**if(br[idle]==p[idle].bTime){**

**p[idle].sTime=currTime;**

**float it=p[idle].sTime-prevT;**

**totalIT+=it;**

**}**

**br[idle]-=1;**

**currTime++;**

**prevT=currTime;**

**if(br[idle]==0){**

**p[idle].cTime=currTime;**

**p[idle].tTime=p[idle].cTime-p[idle].aTime;**

**p[idle].wTime=p[idle].tTime-p[idle].bTime;**

**totalTT+=p[idle].tTime;**

**totalWT+=p[idle].wTime;**

**isCompleted[idle]=1;**

**compTime++;**

**}**

**}**

**else{**

**currTime++;**

**}**

**}**

**float avgTT=(float)totalTT/n;**

**float avgWT=(float)totalWT/n;**

**for(int i=0;i<n-1;i++)**

**{**

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

**{**

**if(p[j].PID>p[j+1].PID)**

**{**

**int temp1=p[j].PID;**

**p[j].PID=p[j+1].PID;**

**p[j+1].PID=temp1;**

**int temp2=p[j].tTime;**

**p[j].tTime=p[j+1].tTime;**

**p[j+1].tTime=temp2;**

**int temp3=p[j].wTime;**

**p[j].wTime=p[j+1].wTime;**

**p[j+1].wTime=temp3;**

**int temp4=p[j].aTime;**

**p[j].aTime=p[j+1].aTime;**

**p[j+1].aTime=temp4;**

**int temp5=p[j].bTime;**

**p[j].bTime=p[j+1].bTime;**

**p[j+1].bTime=temp5;**

**}**

**}**

**}**

**cout<<"Process"<<" "<<"Arrival Time"<<" "<<"Burst Time"<<" "<<"Turnaround Time"<<" "<<"Waiting Time"<<endl;**

**for(int i=0;i<n;i++){**

**cout<<setw(4)<<"P"<<i+1<<" "<<p[i].aTime<<" "<<p[i].bTime<<" "<<p[i].tTime<<" "<<p[i].wTime<<endl;**

**}**

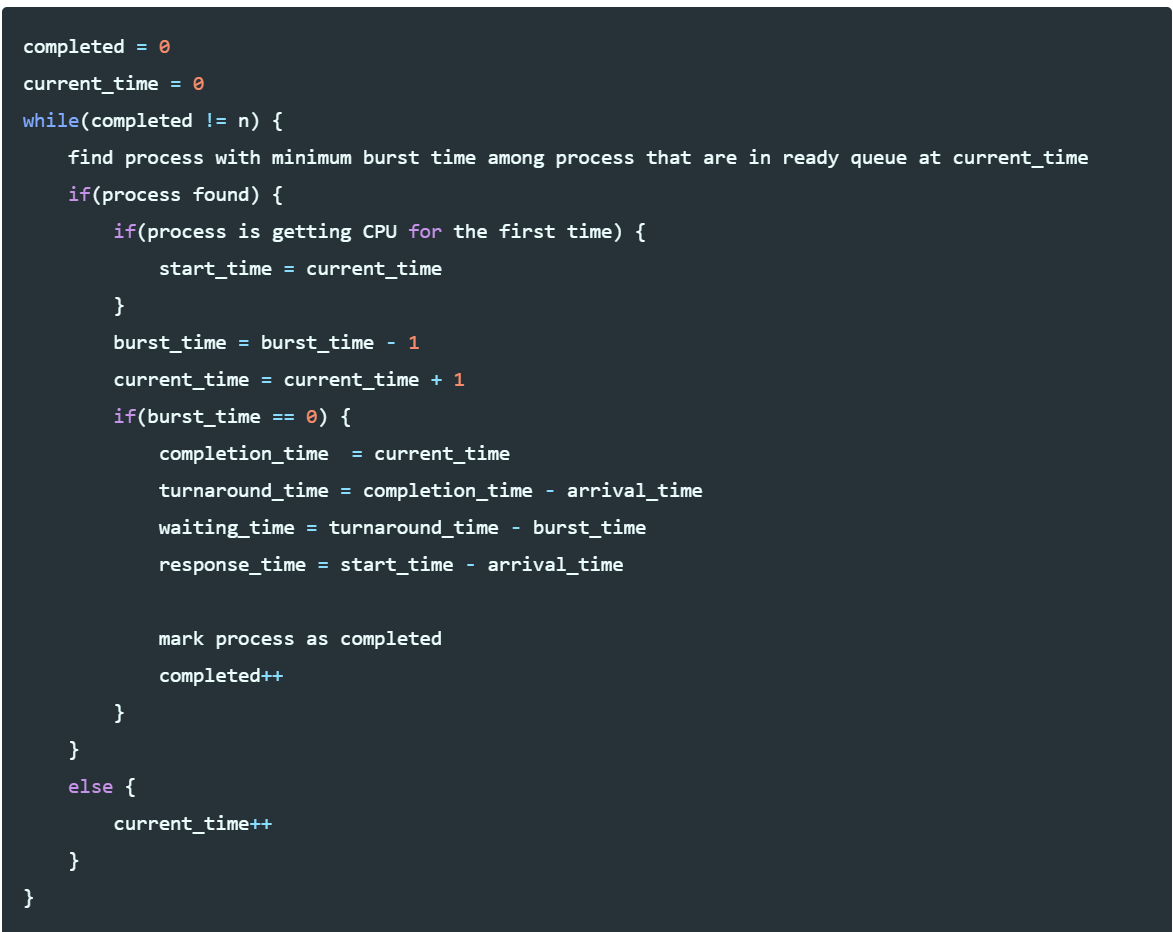
**cout<<"The average turnaround time is = "<<avgTT<<endl;**

**cout<<"The average waiting time is = "<<avgWT<<endl;**

**return 0;**

**}**

**Note:** Algorithm behind it is:



***Priority Scheduling (Non - Pre-emptive)***

**#include <iostream>**

**#include <iomanip>**

**#include <cstring>**

**#include <cmath>**

**using namespace std;**

**class NPPriority{**

**int n;**

**int aTime[50];**

**int bTime[50];**

**int cTime[50];**

**int PID[50];**

**int tTime[50];**

**int wTime[50];**

**int sTime[50];**

**int priority[50];**

**int totalTT;**

**int totalWT;**

**float avgTT;**

**float avgWT;**

**public:**

**NPPriority(){**

**totalTT=0;**

**totalWT=0;**

**}**

**void resort(){**

**for(int i=0;i<n-1;i++)**

**{**

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

**{**

**if(PID[j]>PID[j+1])**

**{**

**int temp1=PID[j];**

**PID[j]=PID[j+1];**

**PID[j+1]=temp1;**

**int temp2=tTime[j];**

**tTime[j]=tTime[j+1];**

**tTime[j+1]=temp2;**

**int temp3=wTime[j];**

**wTime[j]=wTime[j+1];**

**wTime[j+1]=temp3;**

**int temp4=aTime[j];**

**aTime[j]=aTime[j+1];**

**aTime[j+1]=temp4;**

**int temp5=bTime[j];**

**bTime[j]=bTime[j+1];**

**bTime[j+1]=temp5;**

**int temp6=priority[j];**

**priority[j]=priority[j+1];**

**priority[j+1]=temp6;**

**}**

**}**

**}**

**}**

**void findAvgTTAndAvgWT(){**

**int currTime = 0;**

**int complete = 0;**

**int prevTime=0;**

**int isCompleted[50];**

**memset(isCompleted,0,sizeof(isCompleted));**

**while(complete!=n){**

**int ind = -1;**

**int mx = 100000;**

**for(int i=0;i<n;i++){**

**if(currTime>=aTime[i] && isCompleted[i]==0){**

**if(priority[i]<mx){**

**mx=priority[i];**

**ind=i;**

**}**

**if(priority[i]==mx){**

**if(aTime[i]<aTime[ind]){**

**mx=priority[i];**

**ind=i;**

**}**

**}**

**}**

**}**

**if(ind!=-1){**

**sTime[ind]=currTime;**

**cTime[ind]=sTime[ind]+bTime[ind];**

**tTime[ind]=abs(cTime[ind]-aTime[ind]);**

**wTime[ind]=abs(tTime[ind]-bTime[ind]);**

**totalTT+=tTime[ind];**

**totalWT+=wTime[ind];**

**isCompleted[ind]=1;**

**complete++;**

**currTime=cTime[ind];**

**prevTime=currTime;**

**}**

**else{**

**currTime++;**

**}**

**}**

**avgTT=(float)totalTT/n;**

**avgWT=(float)totalWT/n;**

**}**

**void sorting(){**

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

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

**if(aTime[j]>aTime[j+1]){**

**int temp1=aTime[j];**

**aTime[j]=aTime[j+1];**

**aTime[j+1]=temp1;**

**int temp2=bTime[j];**

**bTime[j]=bTime[j+1];**

**bTime[j+1]=temp2;**

**int temp3=priority[j];**

**priority[j]=priority[j+1];**

**priority[j+1]=temp3;**

**int temp4=PID[j];**

**PID[j]=PID[j+1];**

**PID[j+1]=temp4;**

**}**

**}**

**}**

**}**

**void getData(){**

**cout<<"Enter the no of processes: ";**

**cin>>n;**

**cout<<"Enter the burst time: ";**

**for(int i=0;i<n;i++){**

**cin>>bTime[i];**

**}**

**cout<<"Enter the arrival time: ";**

**for(int i=0;i<n;i++){**

**cin>>aTime[i];**

**}**

**cout<<"Enter the priority: ";**

**for(int i=0;i<n;i++){**

**cin>>priority[i];**

**}**

**for(int i=0;i<n;i++){**

**PID[i]=i+1;**

**}**

**}**

**void showData(){**

**sorting();**

**findAvgTTAndAvgWT();**

**resort();**

**cout<<"Process"<<" "<<"Arrival Time"<<" "<<"Burst Time"<<" "<<"Priority"<<" "<<"Turnaround Time"<<" "<<"Waiting Time"<<endl;**

**for(int i=0;i<n;i++){**

**cout<<setw(4)<<"P"<<i+1<<" "<<aTime[i]<<" "<<bTime[i]<<" "<<priority[i]<<" "<<tTime[i]<<" "<<wTime[i]<<endl;**

**}**

**cout<<"The average turnaround time is = "<<avgTT<<endl;**

**cout<<"The average waiting time is = "<<avgWT<<endl;**

**}**

**};**

**int main(){**

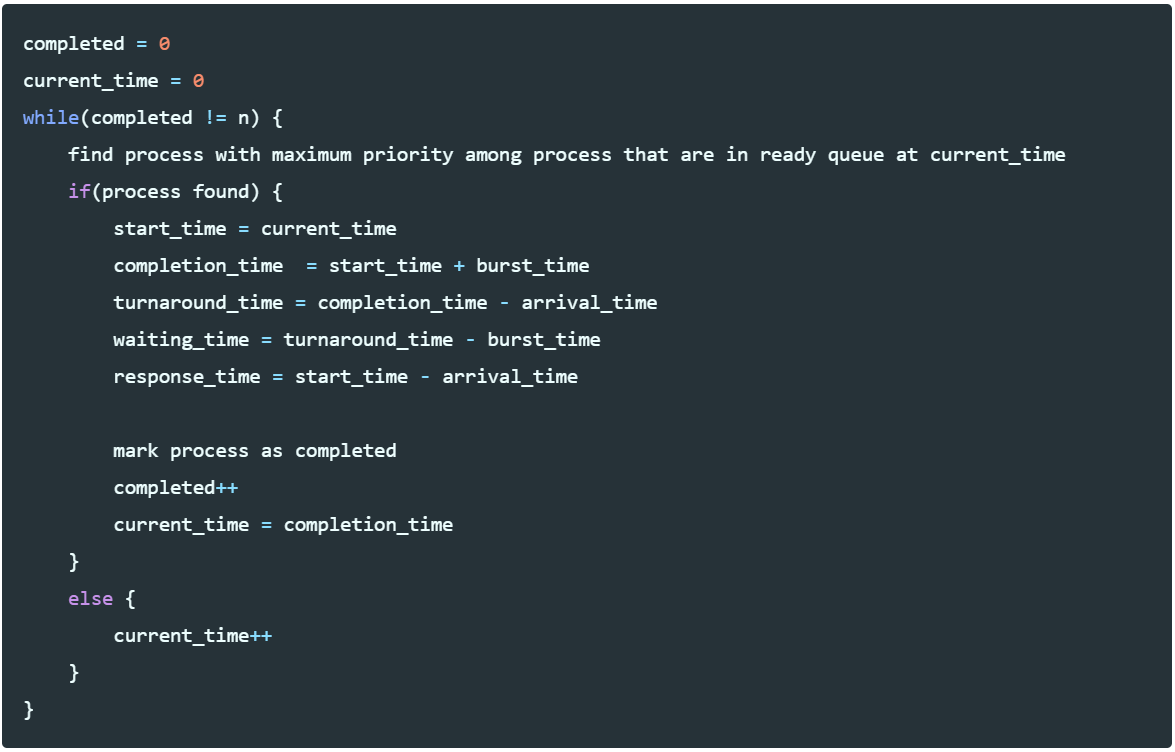
**NPPriority p;**

**p.getData();**

**p.showData();**

**return 0;**

**}**

******

***Priority Scheduling (Pre-emptive)***

**#include <iostream>**

**#include <iomanip>**

**#include <cstring>**

**#include <cmath>**

**using namespace std;**

**class PPriority{**

**int n;**

**int aTime[50];**

**int bTime[50];**

**int cTime[50];**

**int PID[50];**

**int tTime[50];**

**int wTime[50];**

**int sTime[50];**

**int priority[50];**

**int totalTT;**

**int totalWT;**

**int totalIT;**

**float avgTT;**

**float avgWT;**

**public:**

**PPriority(){**

**totalTT=0;**

**totalWT=0;**

**}**

**void resort(){**

**for(int i=0;i<n-1;i++)**

**{**

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

**{**

**if(PID[j]>PID[j+1])**

**{**

**int temp1=PID[j];**

**PID[j]=PID[j+1];**

**PID[j+1]=temp1;**

**int temp2=tTime[j];**

**tTime[j]=tTime[j+1];**

**tTime[j+1]=temp2;**

**int temp3=wTime[j];**

**wTime[j]=wTime[j+1];**

**wTime[j+1]=temp3;**

**int temp4=aTime[j];**

**aTime[j]=aTime[j+1];**

**aTime[j+1]=temp4;**

**int temp5=bTime[j];**

**bTime[j]=bTime[j+1];**

**bTime[j+1]=temp5;**

**int temp6=priority[j];**

**priority[j]=priority[j+1];**

**priority[j+1]=temp6;**

**}**

**}**

**}**

**}**

**void findAvgTTAndAvgWT(){**

**int currTime = 0;**

**int complete = 0;**

**int prevTime=0;**

**int isCompleted[50];**

**memset(isCompleted,0,sizeof(isCompleted));**

**int br[50];**

**for(int i=0;i<n;i++){**

**br[i]=bTime[i];**

**}**

**while(complete!=n){**

**int ind = -1;**

**int mx = 100000;**

**for(int i=0;i<n;i++){**

**if(currTime>=aTime[i] && isCompleted[i]==0){**

**if(priority[i]<mx){**

**mx=priority[i];**

**ind=i;**

**}**

**if(priority[i]==mx){**

**if(aTime[i]<aTime[ind]){**

**mx=priority[i];**

**ind=i;**

**}**

**}**

**}**

**}**

**if(ind!=-1){**

**if(br[ind]==bTime[ind]){**

**sTime[ind]=currTime;**

**float it=sTime[ind]-prevTime;**

**totalIT+=it;**

**}**

**br[ind]-=1;**

**currTime++;**

**prevTime=currTime;**

**if(br[ind]==0){**

**cTime[ind]=currTime;**

**tTime[ind]=abs(cTime[ind]-aTime[ind]);**

**wTime[ind]=abs(tTime[ind]-bTime[ind]);**

**totalTT+=tTime[ind];**

**totalWT+=wTime[ind];**

**isCompleted[ind]=1;**

**complete++;**

**}**

**}**

**else{**

**currTime++;**

**}**

**}**

**avgTT=(float)totalTT/n;**

**avgWT=(float)totalWT/n;**

**}**

**void sorting(){**

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

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

**if(aTime[j]>aTime[j+1]){**

**int temp1=aTime[j];**

**aTime[j]=aTime[j+1];**

**aTime[j+1]=temp1;**

**int temp2=bTime[j];**

**bTime[j]=bTime[j+1];**

**bTime[j+1]=temp2;**

**int temp3=priority[j];**

**priority[j]=priority[j+1];**

**priority[j+1]=temp3;**

**int temp4=PID[j];**

**PID[j]=PID[j+1];**

**PID[j+1]=temp4;**

**}**

**}**

**}**

**}**

**void getData(){**

**cout<<"Enter the no of processes: ";**

**cin>>n;**

**cout<<"Enter the burst time: ";**

**for(int i=0;i<n;i++){**

**cin>>bTime[i];**

**}**

**cout<<"Enter the arrival time: ";**

**for(int i=0;i<n;i++){**

**cin>>aTime[i];**

**}**

**cout<<"Enter the priority: ";**

**for(int i=0;i<n;i++){**

**cin>>priority[i];**

**}**

**for(int i=0;i<n;i++){**

**PID[i]=i+1;**

**}**

**}**

**void showData(){**

**sorting();**

**findAvgTTAndAvgWT();**

**resort();**

**cout<<"Process"<<" "<<"Arrival Time"<<" "<<"Burst Time"<<" "<<"Priority"<<" "<<"Turnaround Time"<<" "<<"Waiting Time"<<endl;**

**for(int i=0;i<n;i++){**

**cout<<setw(4)<<"P"<<i+1<<" "<<aTime[i]<<" "<<bTime[i]<<" "<<priority[i]<<" "<<tTime[i]<<" "<<wTime[i]<<endl;**

**}**

**cout<<"The average turnaround time is = "<<avgTT<<endl;**

**cout<<"The average waiting time is = "<<avgWT<<endl;**

**}**

**};**

**int main(){**

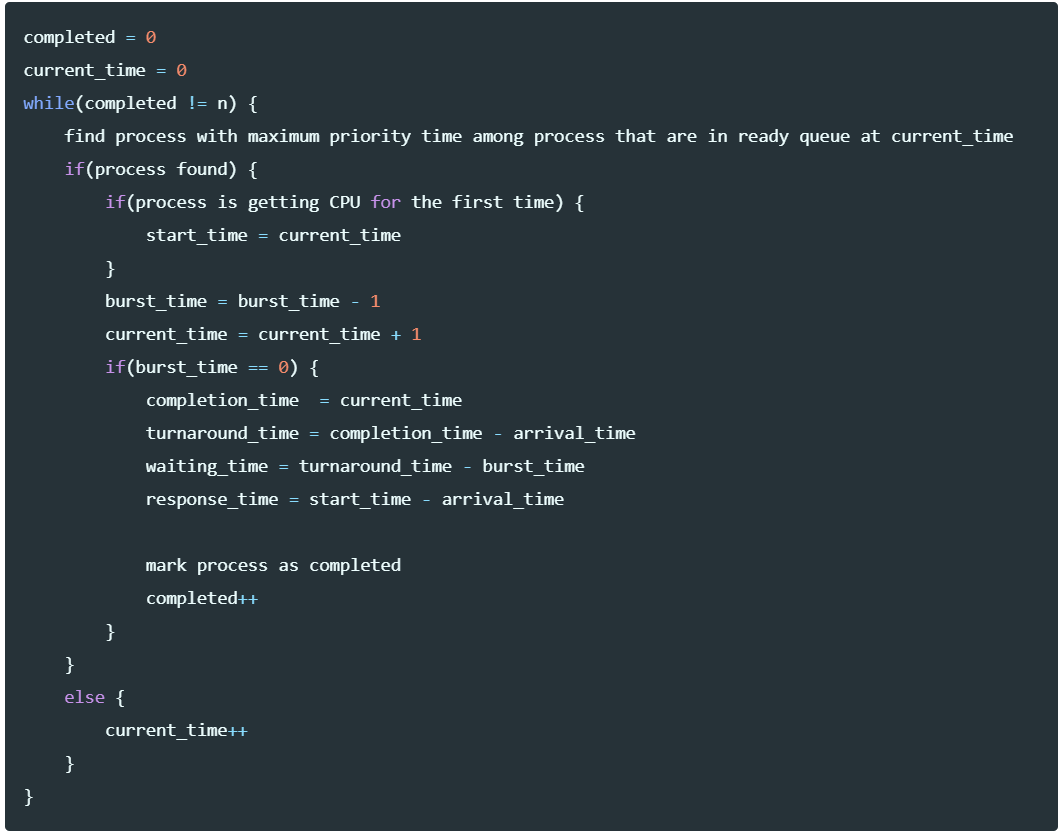
**PPriority p;**

**p.getData();**

**p.showData();**

**return 0;**

**}**

******

***Round Robin Scheduling Algorithm***

**#include<iostream>**

**#include<iomanip>**

**#include<queue>**

**#include<algorithm>**

**using namespace std;**

**struct Process{**

**int PID;**

**float aTime;**

**float bTime;**

**float cTime;**

**float sTime;**

**float tTime;**

**float wTime;**

**};**

**void resort(Process p[50],int n){**

**for(int i=0;i<n-1;i++)**

**{**

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

**{**

**if(p[j].PID>p[j+1].PID)**

**{**

**int temp1=p[j].PID;**

**p[j].PID=p[j+1].PID;**

**p[j+1].PID=temp1;**

**int temp2=p[j].tTime;**

**p[j].tTime=p[j+1].tTime;**

**p[j+1].tTime=temp2;**

**int temp3=p[j].wTime;**

**p[j].wTime=p[j+1].wTime;**

**p[j+1].wTime=temp3;**

**int temp4=p[j].aTime;**

**p[j].aTime=p[j+1].aTime;**

**p[j+1].aTime=temp4;**

**int temp5=p[j].bTime;**

**p[j].bTime=p[j+1].bTime;**

**p[j+1].bTime=temp5;**

**}**

**}**

**}**

**}**

**void sorting(Process p[50],int n){**

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

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

**if(p[j].aTime>p[j+1].aTime){**

**int temp1=p[j].aTime;**

**p[j].aTime=p[j+1].aTime;**

**p[j+1].aTime=temp1;**

**int temp2=p[j].bTime;**

**p[j].bTime=p[j+1].bTime;**

**p[j+1].bTime=temp2;**

**int temp3=p[j].PID;**

**p[j].PID=p[j+1].PID;**

**p[j+1].PID=temp3;**

**}**

**}**

**}**

**}**

**void findAvgTime(Process p[50],int n,float tq){**

**queue <int> q;**

**float br[50];**

**int complete=0;**

**float curr=0;**

**int mark[50];**

**for(int i=0;i<50;i++){**

**mark[i]=0;**

**}**

**sorting(p,n);**

**for(int i=0;i<n;i++){**

**br[i]=p[i].bTime;**

**}**

**float totalTT=0;**

**float totalWT=0;**

**float totalIT=0;**

**q.push(0);**

**int ind=0;**

**mark[0]=0;**

**while(complete!=n){**

**ind=q.front();**

**q.pop();**

**if(br[ind]==p[ind].bTime){**

**p[ind].sTime=max(curr,p[ind].aTime);**

**totalIT+=(p[ind].sTime-curr);**

**curr=p[ind].sTime;**

**}**

**if((br[ind]-tq)>0){**

**br[ind]-=tq;**

**curr+=tq;**

**}**

**else{**

**curr+=br[ind];**

**br[ind]=0;**

**complete++;**

**p[ind].cTime=curr;**

**p[ind].tTime=p[ind].cTime-p[ind].aTime;**

**p[ind].wTime=p[ind].tTime-p[ind].bTime;**

**totalTT+=p[ind].tTime;**

**totalWT+=p[ind].wTime;**

**}**

**for(int i=1;i<n;i++){**

**if(br[i]>0 && p[i].aTime<=curr && mark[i]==0){**

**q.push(i);**

**mark[i]=1;**

**}**

**}**

**if(br[ind]>0){**

**q.push(ind);**

**}**

**if(q.empty()){**

**for(int i=1;i<n;i++){**

**if(br[i]>0){**

**q.push(i);**

**}**

**}**

**}**

**}**

**float avgTT=totalTT/n;**

**float avgWT=totalWT/n;**

**resort(p,n);**

**cout<<"Process"<<" "<<"Arrival Time"<<" "<<"Burst Time"<<" "<<"Turnaround Time"<<" "<<"Waiting Time"<<endl;**

**for(int i=0;i<n;i++){**

**cout<<setw(4)<<"P"<<i+1<<" "<<p[i].aTime<<" "<<p[i].bTime<<" "<<p[i].tTime<<" "<<p[i].wTime<<endl;**

**}**

**cout<<"The average turnaround time is = "<<avgTT<<endl;**

**cout<<"The average waiting time is = "<<avgWT<<endl;**

**}**

**int main(){**

**Process p[50];**

**int n;**

**float tq;**

**cout<<"Enter the no of processes: ";**

**cin>>n;**

**cout<<"Enter the arrival time: ";**

**for(int i=0;i<n;i++){**

**cin>>p[i].aTime;**

**}**

**cout<<"Enter the burst time: ";**

**for(int i=0;i<n;i++){**

**cin>>p[i].bTime;**

**}**

**cout<<"Enter the time quantum: ";**

**cin>>tq;**

**for(int i=0;i<n;i++){**

**p[i].PID=i+1;**

**}**

**findAvgTime(p,n,tq);**

**return 0;**

**}**

**Disk Scheduling Algorithms**

***FCFS Scheduling***

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int main(){**

**int request[51],rcomp[50],moves=0,nr;**

**cout<<"Enter the current position of the pointer:";**

**cin>>request[0];**

**cout<<"Enter the number of pending requests:";**

**cin>>nr;**

**cout<<"Enter the pending request entries:";**

**for(int i=1;i<=nr;i++){**

**cin>>request[i];**

**}**

**for(int i=1,j=i-1;i<=nr;i++,j++){**

**rcomp[j]=abs(request[j]-request[i]);**

**moves+=rcomp[j];**

**}**

**cout<<"Pointer Movement: ";**

**for(int i=0;i<nr;i++){**

**cout<<request[i]<<" -> ";**

**}**

**cout<<request[nr]<<endl;**

**cout<<"Total head movement: "<<moves<<" cylinders."<<endl;**

**}**

***SSTF Scheduling***

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int finds(int a[],int n,int item){**

**for(int i=0;i<=n;i++){**

**if(a[i]==item){**

**return i;**

**}**

**}**

**return -1;**

**}**

**int main(){**

**int request[51],r[51],news[51],comp[51],moves=0,n;**

**cout<<"Enter the current position of the pointer:";**

**cin>>request[0];**

**r[0]=request[0];**

**cout<<"Enter the number of pending requests:";**

**cin>>n;**

**cout<<"Enter the pending request entries:";**

**for(int i=1;i<=n;i++){**

**cin>>request[i];**

**r[i]=request[i];**

**}**

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

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

**if(r[j]>r[j+1]){**

**int temp=r[j];**

**r[j]=r[j+1];**

**r[j+1]=temp;**

**}**

**}**

**}**

**int cp=request[0];**

**int ind=finds(r,n,cp);**

**int i=ind,j=0,k=0,pos1,pos2;**

**news[0]=r[ind];**

**int pos=i;**

**pos1=i+1;**

**pos2=i-1;**

**k++;**

**while(i>0 && i<n){**

**while(r[pos1]==-1){**

**pos1++;**

**}**

**while(r[pos2]==-1){**

**pos2--;**

**}**

**int a=abs(r[pos]-r[pos1]);**

**int b=abs(r[pos]-r[pos2]);**

**r[pos]=-1;**

**if(a>b){**

**comp[j]=b;**

**j++;**

**i=pos2;**

**}**

**else if(a<b){**

**comp[j]=a;**

**j++;**

**i=pos1;**

**}**

**else{**

**int p1=finds(request,n,r[pos1]);**

**int p2=finds(request,n,r[pos2]);**

**if(p1<p2){**

**comp[j]=a;**

**j++;**

**i=pos1;**

**}**

**if(p1>p2){**

**comp[j]=b;**

**j++;**

**i=pos2;**

**}**

**}**

**news[k]=r[i];**

**k++;**

**pos=i;**

**pos1=i+1;**

**pos2=i-1;**

**}**

**if(i==0){**

**int l=1;**

**while(r[l]==-1){**

**l++;**

**}**

**for(int p=l;p<=n;p++){**

**news[k]=r[p];**

**k++;**

**}**

**comp[j]=abs(r[i]-r[n]);**

**j++;**

**}**

**else{**

**int l=0;**

**while(r[l]!=-1){**

**l++;**

**}**

**for(int p=l-1;p>=0;p--){**

**news[k]=r[p];**

**k++;**

**}**

**comp[j]=abs(r[i]-r[0]);**

**j++;**

**}**

**for(int i=0;i<j;i++){**

**moves+=comp[i];**

**}**

**cout<<"Pointer Movement: ";**

**for(int i=0;i<k-1;i++){**

**cout<<news[i]<<" -> ";**

**}**

**cout<<news[k-1]<<endl;**

**cout<<"Total head movement: "<<moves<<" cylinders."<<endl;**

**}**

***SCAN Scheduling***

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int finds(int a[],int n,int item){**

**for(int i=0;i<=n;i++){**

**if(a[i]==item){**

**return i;**

**}**

**}**

**return -1;**

**}**

**int main(){**

**int request[51],news[51],comp,sizes,n;**

**cout<<"Enter the disk size:";**

**cin>>sizes;**

**cout<<"Enter the current position of the pointer:";**

**cin>>request[0];**

**int cp=request[0];**

**cout<<"Enter the number of pending requests:";**

**cin>>n;**

**cout<<"Enter the pending request entries:";**

**for(int i=1;i<=n;i++){**

**cin>>request[i];**

**}**

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

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

**if(request[j]>request[j+1]){**

**int temp=request[j];**

**request[j]=request[j+1];**

**request[j+1]=temp;**

**}**

**}**

**}**

**int ind=finds(request,n,cp);**

**int i=ind,j=0,pos1,pos2;**

**news[0]=request[ind];**

**pos1=i+1;**

**pos2=i-1;**

**j++;**

**int ctr1=0,ctr2=0;**

**for(int k=pos1;k<=n;k++){**

**ctr1++;**

**}**

**for(int l=pos2;l>=0;l--){**

**ctr2++;**

**}**

**if(ctr1>ctr2){**

**for(int k=pos1;k<=n;k++){**

**news[j]=request[k];**

**j++;**

**}**

**news[j]=sizes-1;**

**j++;**

**for(int l=pos2;l>=0;l--){**

**news[j]=request[l];**

**j++;**

**}**

**news[j]=0;**

**j++;**

**comp=abs(cp-(sizes-1))+(sizes-1);**

**}**

**else{**

**for(int l=pos2;l>=0;l--){**

**news[j]=request[l];**

**j++;**

**}**

**news[j]=0;**

**j++;**

**for(int k=pos1;k<=n;k++){**

**news[j]=request[k];**

**j++;**

**}**

**news[j]=sizes-1;**

**j++;**

**comp=cp+(sizes-1);**

**}**

**cout<<"Pointer Movement: ";**

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

**cout<<news[i]<<" -> ";**

**}**

**cout<<news[j-1]<<endl;**

**cout<<"Total head movement: "<<comp<<" cylinders."<<endl;**

**}**

***C-SCAN Scheduling***

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int finds(int a[],int n,int item){**

**for(int i=0;i<=n;i++){**

**if(a[i]==item){**

**return i;**

**}**

**}**

**return -1;**

**}**

**int main(){**

**int request[51],news[51],comp,sizes,n;**

**cout<<"Enter the disk size:";**

**cin>>sizes;**

**cout<<"Enter the current position of the pointer:";**

**cin>>request[0];**

**int cp=request[0];**

**cout<<"Enter the number of pending requests:";**

**cin>>n;**

**cout<<"Enter the pending request entries:";**

**for(int i=1;i<=n;i++){**

**cin>>request[i];**

**}**

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

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

**if(request[j]>request[j+1]){**

**int temp=request[j];**

**request[j]=request[j+1];**

**request[j+1]=temp;**

**}**

**}**

**}**

**int ind=finds(request,n,cp);**

**int i=ind,j=0,pos1,pos2;**

**news[0]=request[ind];**

**pos1=i+1;**

**pos2=i-1;**

**j++;**

**int ctr1=0,ctr2=0;**

**for(int k=pos1;k<=n;k++){**

**ctr1++;**

**}**

**for(int l=pos2;l>=0;l--){**

**ctr2++;**

**}**

**if(ctr1>ctr2){**

**for(int k=pos1;k<=n;k++){**

**news[j]=request[k];**

**j++;**

**}**

**news[j]=sizes-1;**

**j++;**

**news[j]=0;**

**j++;**

**for(int l=0;l<=pos2;l++){**

**news[j]=request[l];**

**j++;**

**}**

**comp=abs(cp-(sizes-1))+(sizes-1)+request[pos2];**

**}**

**else{**

**for(int l=pos2;l>=0;l--){**

**news[j]=request[l];**

**j++;**

**}**

**news[j]=0;**

**j++;**

**news[j]=sizes-1;**

**j++;**

**for(int k=n;k>=pos1;k--){**

**news[j]=request[k];**

**j++;**

**}**

**comp=cp+(sizes-1)+((sizes-1)-request[pos1]);**

**}**

**cout<<"Pointer Movement: ";**

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

**cout<<news[i]<<" -> ";**

**}**

**cout<<news[j-1]<<endl;**

**cout<<"Total head movement: "<<comp<<" cylinders."<<endl;**

**}**

***LOOK Scheduling***

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int finds(int a[],int n,int item){**

**for(int i=0;i<=n;i++){**

**if(a[i]==item){**

**return i;**

**}**

**}**

**return -1;**

**}**

**int main(){**

**int request[51],news[51],comp,n;**

**cout<<"Enter the current position of the pointer:";**

**cin>>request[0];**

**int cp=request[0];**

**cout<<"Enter the number of pending requests:";**

**cin>>n;**

**cout<<"Enter the pending request entries:";**

**for(int i=1;i<=n;i++){**

**cin>>request[i];**

**}**

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

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

**if(request[j]>request[j+1]){**

**int temp=request[j];**

**request[j]=request[j+1];**

**request[j+1]=temp;**

**}**

**}**

**}**

**int ind=finds(request,n,cp);**

**int i=ind,j=0,pos1,pos2;**

**news[0]=request[ind];**

**pos1=i+1;**

**pos2=i-1;**

**j++;**

**int ctr1=0,ctr2=0;**

**for(int k=pos1;k<=n;k++){**

**ctr1++;**

**}**

**for(int l=pos2;l>=0;l--){**

**ctr2++;**

**}**

**if(ctr1>ctr2){**

**for(int k=pos1;k<=n;k++){**

**news[j]=request[k];**

**j++;**

**}**

**for(int l=pos2;l>=0;l--){**

**news[j]=request[l];**

**j++;**

**}**

**comp=abs(cp-request[n])+abs(request[0]-request[n]);**

**}**

**else{**

**for(int l=pos2;l>=0;l--){**

**news[j]=request[l];**

**j++;**

**}**

**for(int k=pos1;k<=n;k++){**

**news[j]=request[k];**

**j++;**

**}**

**comp=abs(cp-request[0])+abs(request[0]-request[n]);**

**}**

**cout<<"Pointer Movement: ";**

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

**cout<<news[i]<<" -> ";**

**}**

**cout<<news[j-1]<<endl;**

**cout<<"Total head movement: "<<comp<<" cylinders."<<endl;**

**}**

***C-LOOK Scheduling***

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int finds(int a[],int n,int item){**

**for(int i=0;i<=n;i++){**

**if(a[i]==item){**

**return i;**

**}**

**}**

**return -1;**

**}**

**int main(){**

**int request[51],news[51],comp,n;**

**cout<<"Enter the current position of the pointer:";**

**cin>>request[0];**

**int cp=request[0];**

**cout<<"Enter the number of pending requests:";**

**cin>>n;**

**cout<<"Enter the pending request entries:";**

**for(int i=1;i<=n;i++){**

**cin>>request[i];**

**}**

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

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

**if(request[j]>request[j+1]){**

**int temp=request[j];**

**request[j]=request[j+1];**

**request[j+1]=temp;**

**}**

**}**

**}**

**int ind=finds(request,n,cp);**

**int i=ind,j=0,pos1,pos2;**

**news[0]=request[ind];**

**pos1=i+1;**

**pos2=i-1;**

**j++;**

**int ctr1=0,ctr2=0;**

**for(int k=pos1;k<=n;k++){**

**ctr1++;**

**}**

**for(int l=pos2;l>=0;l--){**

**ctr2++;**

**}**

**if(ctr1>ctr2){**

**for(int k=pos1;k<=n;k++){**

**news[j]=request[k];**

**j++;**

**}**

**for(int l=0;l<=pos2;l++){**

**news[j]=request[l];**

**j++;**

**}**

**comp=abs(cp-request[n])+(request[n]-request[0])+abs(request[0]-request[pos2]);**

**}**

**else{**

**for(int l=pos2;l>=0;l--){**

**news[j]=request[l];**

**j++;**

**}**

**for(int k=n;k>=pos1;k--){**

**news[j]=request[k];**

**j++;**

**}**

**comp=abs(cp-request[0])+abs(request[0]-request[n])+abs(request[pos1]-request[n]);**

**}**

**cout<<"Pointer Movement: ";**

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

**cout<<news[i]<<" -> ";**

**}**

**cout<<news[j-1]<<endl;**

**cout<<"Total head movement: "<<comp<<" cylinders."<<endl;**

**}**