// FCFS

#include<stdio.h>

**int** main()

{

**int** p,bt[10],at[10],tat[10],wt[10],i,j,temp[10];

**float** awt=0,atat=0;

printf("enter the no of process\n");

scanf("%d",&p);

printf("enter the burst time\n");

**for**(i=0;i<p;i++)

{

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

}

printf("process \t burst time \t waiting time \t turn around time \n");

**for**(**int** i=0;i<p;i++)

{

wt[i]=0;

tat[i]=0;

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

{

wt[i]=wt[i]+bt[j];

}

tat[i]=wt[i]+bt[i];

awt=awt+wt[i];

atat=atat+tat[i];

printf("\n%d \t\t%d \t\t%d \t\t%d",i+1,bt[i],wt[i],tat[i]);

}

awt=awt/p;

atat=atat/p;

printf("average waiting time is %f",awt);

printf("average turn around time is %f",atat);

**return** 0;

}

//SJF

#include<stdio.h>

**int** main()

{

**int** n,p[10],bt[10],tt[10],wt[10],i,j,t;

**float** awt=0,atat=0;

printf("enter the no of process");

scanf("%d",&n);

printf("enter the process number");

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

{

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

}

printf("enter the burst time");

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

{

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

}

//applying sorting

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

{

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

{

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

{

t=bt[j];

bt[j]=bt[j+1];

bt[j+1]=t;

t=p[j];

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

p[j+1]=t;

}

}

}

printf("process \t burst time \t waiting time \t turnaround time \n");

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

{

wt[i]=0;

tt[i]=0;

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

{

wt[i]=wt[i]+bt[j];

}

tt[i]=wt[i]+bt[i];

awt=awt+wt[i];

atat=atat+wt[i];

printf("\n%d\t\t%d\t\t%d\t\t%d /n",i+1,bt[i],wt[i],tt[i]);

}

awt=awt/n;

atat=atat/n;

printf("average waiting time is %f",awt);

printf("average turn around time is %f",atat);

**return** 0;

}

//priority

#include<stdio.h>

**int** main()

{

**int** i,j,bt[10],wt[10],tat[10],n,pr[10],pos,temp;

**float** awt=0, atat=0;

printf("enter the no of process");

scanf("%d",&n);

printf("enter the burst time");

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

{

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

}

printf("enter the priority of the process");

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

{

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

}

//applying sorting

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

{

pos=i;

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

{

**if**(pr[j]<pr[pos])

{

pos=j;

}

}

temp=pr[i];

pr[i]=pr[pos];

pr[pos]=temp;

temp=bt[i];

bt[i]=bt[pos];

bt[pos]=temp;

}

wt[0]=0;

printf("process\tbursttime\tpriority\twaitingtime\turnaroundtime\n");

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

{

wt[i]=0;

tat[i]=0;

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

{

wt[i]=wt[i]+bt[j];

}

tat[i]=wt[i]+bt[i];

awt=awt+wt[i];

atat=atat+tat[i];

printf("\n %d\t\t%d\t\t%d\t\t%d\t\t%d\n",i+1,bt[i],pr[i],wt[i],tat[i]);

}

awt=awt/n;

atat=atat/n;

printf("average waiting time is %f\n",awt);

printf("average turn ariund time is %f\n",atat);

**return** 0;

}

//round robin

#include<stdio.h>

**int** main()

{

**int** n,qt, bt[10],wt[10],tat[10],count=0,rem\_bt[10],i,temp;

**int** sq=0;

**float** awt=0,atat=0;

printf("enter the no of the process:\n");

scanf("%d",&n);

printf("enter thr burst time of the process:\n");

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

{

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

rem\_bt[i]=bt[i];

}

printf("enter the quantum time");

scanf("%d",&qt);

**while**(1)

{

**for**(i=0,count=0;i<n;i++)

{

temp=qt;

**if**(rem\_bt[i]==0)

{

count++;

**continue**;

}

**if**(rem\_bt[i]>qt)

{

rem\_bt[i]=rem\_bt[i]-qt;

}

**else** **if**(rem\_bt[i]>0)

{

temp=rem\_bt[i];

rem\_bt[i]=0;

}

sq=sq+temp;

tat[i]=sq;

}

**if**(n==count)

**break**;

}

printf("\nprocess\tburst time\tturnaround time\twaiting time\n");

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

{

wt[i]=tat[i]-bt[i];

awt=awt+wt[i];

atat=atat+tat[i];

printf("\n%d\t%d\t\t%d\t\t%d",i+1,bt[i],tat[i],wt[i]);

}

awt=awt/n;

atat=atat/n;

printf("average the waiting time is %f",awt);

printf("average turn around time is %f",atat);

**return** 0;

}

//first fit

#include<stdio.h>

**int** main(){

**int** h,p,holes[20],process[20],ho[20],i,j,flag=0;

printf("Enter the number of holes:");

scanf("%d",&h);

**for**(i=0;i<h;i++){

printf("Enter the size of hole H%d:",i+1);

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

ho[i]=holes[i];

}

printf("Enter the number of processes:");

scanf("%d",&p);

**for**(i=0;i<p;i++){

printf("Enter the size of process P%d:",i+1);

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

}

printf("Process\tpsize\thole\n");

**for**(i=0;i<p;i++){

flag=0;

**for**(j=0;j<h;j++){

**if**(holes[j]>=process[i]){

holes[j]=holes[j]-process[i];

printf("P%d\t%d\tH%d\n",i+1,process[i],j+1);

flag=1;

**break**;

}

}

**if**(flag==0)

printf("P%d\t%d\tNot allocated\n",i+1,process[i]);

}

printf("Space available after allocation\n");

printf("hole\tactual\tavailable\n");

**for**(i=0;i<h;i++)

printf("H%d\t%d\t%d\n",i+1,ho[i],holes[i]);

**return** 0;

}

//best fit

#include<stdio.h>

**void** sort();

**int** main(){

**int** h,p,holes[20],parray[20],ho[20],process[20],i,j,flag=0;

printf("Enter the number of holes:");

scanf("%d",&h);

**for**(i=0;i<h;i++){

printf("Enter the size of hole H%d:",i+1);

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

ho[i]=holes[i];

parray[i]=holes[i];

}

printf("Enter the number of processes:");

scanf("%d",&p);

**for**(i=0;i<p;i++){

printf("Enter the size of process P%d:",i+1);

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

}

printf("Process\tpsize\thole\n");

**for**(i=0;i<p;i++){

flag=0;

sort(h,holes);

**for**(j=0;j<h;j++){

**if**(holes[j]>=process[i]){

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

**if**(parray[k]==holes[j]){

holes[j]=holes[j]-process[i];

parray[k]=parray[k]-process[i];

printf("P%d\t%d\tH%d\n",i+1,process[i],k+1);

flag=1;

**break**;

}

}

**break**;

}

}

**if**(flag==0)

printf("P%d\t%d\tNot allocated\n",i+1,process[i]);

}

printf("Space available after allocation\n");

printf("hole\tactual\tavailable\n");

**for**(i=0;i<h;i++){

printf("H%d\t%d\t%d\n",i+1,ho[i],parray[i]);

}

**return** 0;

}

**void** sort(**int** h, **int** holes[20]){

**int** swap;

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

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

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

swap=holes[j];

holes[j]=holes[j+1];

holes[j+1]=swap;

}

}

}

}

//worst fit

#include<stdio.h>

**void** sort(**int** h, **int** holes[10])

{

**int** swap;

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

{

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

{

**if**(holes[j]<holes[j+1])

{

swap=holes[j];

holes[j]=holes[j+1];

holes[j+1]=swap;

}

}

}

}

**int** main()

{

**int** h,p,holes[10],process[10],actual[10],parallel[10],i,j,k,flag=0;

printf("enter the no of holes");

scanf("%d",&h);

**for**(i=0;i<h;i++)

{

printf("enter the hole size h%d",i+1);

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

actual[i]=holes[i];

parallel[i]=holes[i];

}

printf("enter the no of process");

scanf("%d",&p);

**for**(i=0;i<p;i++)

{

printf("enter the process size p%d",i+1);

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

}

printf("\nprocess\tpsize\tholes\n");

**for**(i=0;i<p;i++)

{

flag=0;

sort(h,holes);

**for**(j=0;j<h;j++)

{

**if**(holes[j]>=process[i])

{

**for**(k=0;k<h;k++)

{

**if**(parallel[k]==holes[j])

{

holes[j]=holes[j]-process[i];

parallel[k]=parallel[k]-process[i];

printf("\n%d\t%d\t%d\n",i+1,process[i],j+1);

flag=1;

**break**;

}

}

**break**;

}

}

**if**(flag==0)

{

printf("\n%d\t%d\tnot allocated",i+1,process[i]);

}

}

printf("available spaces");

printf("\nholes\tactual\tavailable\n");

**for**(i=0;i<h;i++)

{

printf("\n%d\t%d\t%d",i+1,actual[i],parallel[i]);

}

**return** 0;

}

//fifo

#include <stdio.h>

**int** main() {

**int** n, pg[10], frame, fr[10], hit = 0, k = 0, fault = 0, flag = 0, i, j;

printf("Enter the number of elements: ");

scanf("%d", &n);

printf("Enter the number of pages:\n");

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

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

}

printf("Enter the number of frames: ");

scanf("%d", &frame);

**for** (i = 0; i < frame; i++) {

fr[i] = -1;

}

printf("Reference String\tPage Frames\n");

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

flag = 0;

**for** (j = 0; j < frame; j++) {

**if** (fr[j] == pg[i]) {

flag = 1;

hit++;

}

}

**if** (flag == 0) {

fr[k] = pg[i];

k = (k + 1) % frame;

fault++;

}

printf("%d\t\t", pg[i]);

**for** (j = 0; j < frame; j++) {

printf("%d ", fr[j]);

}

printf("\n");

}

printf("\nTotal number of faults: %d\n", fault);

printf("Total number of hits: %d\n", hit);

printf("\nLast Frame:\n");

**for** (i = 0; i < frame; i++) {

printf("%d\t", fr[i]);

}

printf("\n");

**return** 0;

}

//fscan

#include<stdio.h>

int main(){

int request\_arr[20],request,head,seek=0,i;

printf("Enter the number of repuests:");

scanf("%d",&request);

printf("Enter the requests:\n");

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

scanf("%d",&request\_arr[i]);

printf("Enter the initial head position:");

scanf("%d",&head);

printf("%d->",head);

for(i=0;i<request;i++){

if(request\_arr[i]>head)

seek+=(request\_arr[i]-head);

else

seek+=(head-request\_arr[i]);

head=request\_arr[i];

}

for(i=0;i<request-1;i++)

printf("%d->",request\_arr[i]);

printf("%d",request\_arr[i]);

printf("\nTotal seek distance is:%d\n",seek);

return 0;

}

//scan

#include<stdio.h>

int request\_arr[20],request,head,size,seek=0,i,j,direction;

int main(){

printf("Enter the number of requests:");

scanf("%d",&request);

printf("Enter the requests:\n");

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

scanf("%d",&request\_arr[i]);

printf("Enter the initial head position:");

scanf("%d",&head);

printf("Total disk size:");

scanf("%d",&size);

printf("Enter the head movement direction\n0-From the beginning\n1-From the end track\n");

scanf("%d",&direction);

//sort

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

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

int temp;

if(request\_arr[j]>request\_arr[j+1]){

temp=request\_arr[j];

request\_arr[j]=request\_arr[j+1];

request\_arr[j+1]=temp;

}

}

}

printf("%d->",head);

//case 1

if(direction==0){

int start=0;

for(i=0;i<request;i++){

if(request\_arr[i]<head)

start=i;

else

break;

}

for(i=start;i>=0;i--){

seek+=(head-request\_arr[i]);

head=request\_arr[i];

printf("%d->",request\_arr[i]);

}

seek+=head;

printf("0->");

head=0;

//greater than head

for(i=start+1;i<request-1;i++){

seek+=(request\_arr[i]-head);

head=request\_arr[i];

printf("%d->",request\_arr[i]);

}

seek+=request\_arr[i]-head;

printf("%d",request\_arr[i]);

}

//case 2

else{

int start=0;

for(i=0;i<request;i++){

if(request\_arr[i]<head)

start=i;

else

break;

}

for(i=start+1;i<request;i++){

seek+=(request\_arr[i]-head);

head=request\_arr[i];

printf("%d->",request\_arr[i]);

}

printf("%d->",size-1);

seek+=((size-1)-head);

head=size-1;

for(i=start;i>0;i--){

seek+=(head-request\_arr[i]);

head=request\_arr[i];

printf("%d->",request\_arr[i]);

}

seek+=(head-request\_arr[i]);

printf("%d\n",request\_arr[i]);

}

printf("\nTotal seek distance is:%d\n",seek);

return 0;

}