**OPERATING SYSTEM ------- ELAB ANSWERS**

**SUB: OPERATING SYSTEM**  **SUB CODE : USA20302J**

**BCA : ‘ A ‘ , Staff : Mrs.V.Devi By: Syugendran**

**ATTENTION : Before doing program check which programing language that your going to use , choose and run that language in elab.**

**SESSION : 1 ;**

Q. 1: Gantt Chart 1

QUESTION DESCRIPTION  
  
Software Engineer Saran is assigned with several projects from various companies by his Team lead. He is also insisted to complete the projects based on their duration (ie.,Shortest duration first).He is asked to submit a chart presentation for the above project. Help him to draw the chart alone.  
INPUT:  
1st line : number of process  
2nd line : Process ID of each process  
3rd line : Burst time of each process  
4th line : arrival time of each process

TEST CASE 1  
  
INPUT

3  
4 5 6  
8 4 5  
1 2 3

OUTPUT

The gantt chart  
P5 P6 P4   
0 4 9 17

**PROGRAM :**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int n,yug;**

**int temp;**

**cin>>n;**

**int p[n] , b[n],a[n],c[n];**

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

**{**

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

**}**

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

**{**

**cin>>b[i];**

**}**

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

**{**

**cin>>a[i];**

**}**

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

**{**

**yug=i;**

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

**{**

**if(b[j]<b[yug])**

**yug=j;**

**}**

**temp=b[i];**

**b[i]=b[yug];**

**b[yug]=temp;**

**temp=p[i];**

**p[i]=p[yug];**

**p[yug]=temp;**

**}**

**cout<<"The gantt chart\n";**

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

**{**

**cout<<"P"<<p[i]<<" ";**

**}**

**c[0]=0;**

**cout<<endl;**

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

**{**

**c[i]=0;**

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

**{**

**c[i]+=b[j];**

**}**

**}**

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

**{**

**cout<<c[i]<<" ";**

**}**

**return 0;**

**}**

Q. 2: Queue in Airport

QUESTION DESCRIPTION  
  
Semester break has started and Ashwin is going home from College (SRM UNIVERSITY). He is taking flight from Chennai to Delhi. After entering the Airport, he was asked to get into queue of the Boarding Pass.  
  
The Airline staff (Jet Airways) follows the Multi Level Feedback Queue Scheduling algorithm (The algorithm allocates the jobs or processes to different queues based on their CPU execution time. If a process has a large burst-time, then it is automatically moved to a lower-priority queue. )  
  
Ram is a programmer and wanted to gift a program to the airline which shows the starting time the person enter the queue and the end time until which he is serviced.  
  
Input:  
The first line contains the Number of Passengers n. The next n line contains the Process id,,arrival time,burst time and the process priority number of that passenger.  
  
Output:  
The first n lines should print Process Number Starting time when the passenger is started servicing and the time when his job is finished.

TEST CASE 1  
  
INPUT

5  
1 4 10 2  
2 7 48 6  
3 2 64 1  
4 8 32 5  
5 5 50 3

OUTPUT

P1 Time:10 To 20  
P5 Time:20 To 70  
P3 Time:70 To 134  
P4 Time:134 To 166  
P2 Time:166 To 214

**PROGRAM:**

**#include<iostream>**

**using namespace std;**

**#define N 10**

**int Queue(int t1)**

**{**

**if(t1 == 0 || t1 == 1 || t1 == 2 || t1 == 3)**

**{**

**return 1;**

**}**

**else**

**{**

**return 2;**

**}**

**}**

**typedef struct**

**{**

**int process\_id, arrival\_time, burst\_time, priority;**

**int q, ready;**

**}process\_structure;**

**int main()**

**{**

**int limit, count, temp\_process, time, j, y;**

**process\_structure temp;**

**scanf("%d", &limit);**

**process\_structure process[limit];**

**for(count = 0; count < limit; count++)**

**{**

**cin>>process[count].process\_id;**

**cin>>process[count].arrival\_time;**

**cin>>process[count].burst\_time;**

**cin>>process[count].priority;**

**temp\_process = process[count].priority;**

**process[count].q = Queue(temp\_process);**

**process[count].ready = 0;**

**}**

**time = process[0].burst\_time;**

**for(y = 0; y < limit; y++)**

**{**

**for(count = y; count < limit; count++)**

**{**

**if(process[count].arrival\_time < time)**

**{**

**process[count].ready = 1;**

**}**

**}**

**for(count = y; count < limit - 1; count++)**

**{**

**for(j = count + 1; j < limit; j++)**

**{**

**if(process[count].ready == 1 && process[j].ready == 1)**

**{**

**if(process[count].q == 2 && process[j].q == 1)**

**{**

**temp = process[count];**

**process[count] = process[j];**

**process[j] = temp;**

**}**

**}**

**}**

**}**

**for(count = y; count < limit - 1; count++)**

**{**

**for(j = count + 1; j < limit; j++)**

**{**

**if(process[count].ready == 1 && process[j].ready == 1)**

**{**

**if(process[count].q == 1 && process[j].q == 1)**

**{**

**if(process[count].burst\_time > process[j].burst\_time)**

**{**

**temp = process[count];**

**process[count] = process[j];**

**process[j] = temp;**

**}**

**else**

**{**

**break;**

**}**

**}**

**}**

**}**

**}**

**cout<<"\nP"<<process[y].process\_id<< " Time:"<<time<<" To "<<time + process[y].burst\_time;**

**time = time + process[y].burst\_time;**

**for(count = y; count < limit; count++)**

**{**

**if(process[count].ready == 1)**

**{**

**process[count].ready = 0;**

**}**

**}**

**}**

**return 0;**

**}**

Q. 3: Semester Break

QUESTION DESCRIPTION  
  
Semester break has started and Bogar is going home from College (Tamil Siddha). He is taking flight from Chennai to Delhi. After entering the Airport, he was asked to get into queue of the Boarding Pass. The Airline staff follows the FCFS algorithm and first process the request of the person who came first that is who is at first position in the queue.  
  
Bogar is a programmer and wanted to gift a program to the airline which shows the service time (time at which that person will reach the counter), waiting time (service time -arrival time) for each person in the queue and turnaround time (total time from arrival till person gets boarding pass). Also, Calculate average waiting and average turnaround time.  
  
Input: The first line contains the Number of Passengers n. The next n line contains the Burst time and Arrival time.  
  
Output: - The first n lines should print input values, service time (time at which that person will reach the counter), waiting time (service time -arrival time) for each person in the queue and turnaround time (total time from arrival till person gets boarding pass).  
  
In next two lines Print average waiting and average turnaround time in next two lines respectively.

TEST CASE 1  
  
INPUT

4  
4 9 8 3  
0 2 4 3

OUTPUT

Passenger0 0 4  
Passenger1 2 11  
Passenger2 9 17  
Passenger3 18 21  
Average waiting time=7.250000  
Average turn around time=13.250000

**PROGRAM:**

**#include <iostream>**

**#include <stdio.h>**

**using namespace std;**

**int main() {**

**int input\_len , \*pri , \*bt , temp\_wt=0 , temp\_tr=0 , sum\_wt = 0 , sum\_tr = 0;**

**cin>>input\_len;**

**pri = new int[input\_len];**

**bt = new int[input\_len];**

**for(int i =0 ;i<input\_len;i++){**

**cin>>bt[i];**

**}**

**for(int i =0 ;i<input\_len;i++){**

**cin>>pri[i];**

**}**

**for(int i =0 ;i<input\_len;i++){**

**temp\_wt =0;**

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

**temp\_wt+=bt[j];**

**}**

**temp\_wt-=pri[i];**

**temp\_tr = temp\_wt+bt[i];**

**sum\_wt+=temp\_wt;**

**sum\_tr += temp\_tr;**

**cout<<"Passenger"<<i<<' '<<temp\_wt<<' '<<temp\_tr<<endl;**

**}**

**float val = (float)sum\_wt/(float)input\_len;**

**printf("Average waiting time=%.6f\n",val);**

**val = (float)sum\_tr / (float)input\_len;**

**printf("Average turn around time=%.6f",val);**

**return 0;**

**}**

Q. 4: Chirag Flight Travel

QUESTION DESCRIPTION  
  
Semester break has started and Chirag is going home from College (SRM UNIVERSITY). He is taking flight from Chennai to Delhi. After entering the Airport, he was asked to directly go to the counter of boarding pass and without following the queue.  
  
He was Told that The Airline staff follows the SJF (Shortest Job First) algorithm and first process the request of the person who came with the lowest baggage weight as it is easy to process and the waiting time reduces. Chirag is a programmer and wanted to gift a program to the airline which shows the service time (time at which that person will reach the counter), waiting time (service time -arrival time) for each person in the queue and turnaround time (total time from arrival till person gets boarding pass). Also, Calculate average waiting and average turnaround time.  
  
Input: The first line contains the Number of Passengers n. The next n line contains service time.  
  
Output: - The first n lines should print input values, Gnatt Chart for waiting time, Average waiting time, Gnatt Chart for Turn around time, Average turn around time.

TEST CASE 1  
  
INPUT

4  
5 9 7 3

OUTPUT

Gnatt Chart for Waiting time=3 8 15   
Average WT=6  
Gnatt Chart for Turn Around Time=3 8 15 24   
Average TAT=50  
P4 3 0 3  
P1 5 3 8  
P3 7 8 15  
P2 9 15 24

TEST CASE 2  
  
INPUT

5  
6 4 5 3 7

**PROGRAM:**

**// SJF NON PRE**

**#include<stdio.h>**

**int main()**

**{**

**int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;**

**int avg\_wt;//avg\_tat;**

**scanf("%d",&n);**

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

**{**

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

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

**}**

**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;**

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

**{**

**wt[i]=0;**

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

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

**total+=wt[i];**

**}**

**avg\_wt= total/n;**

**total=0;**

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

**{**

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

**total+=tat[i];**

**}**

**printf("Gnatt Chart for Waiting time=");**

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

**{**

**printf("%d ",wt[i]);**

**}**

**printf("\nAverage WT=%d",avg\_wt);**

**printf("\nGnatt Chart for Turn Around Time=");**

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

**{**

**printf("%d ",tat[i]);**

**}**

**printf("\nAverage TAT=%d",total);**

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

**{**

**printf("\nP%d %d %d %d",p[i],bt[i],wt[i],tat[i]);**

**}**

**return 0;**

**}**

Q. 5: Project Duration

QUESTION DESCRIPTION  
  
Software Engineer Saran is assigned with several projects from various companies by his Team lead. He is also insisted to complete the projects based on their duration (ie.,Shortest duration first). Project order date and their duration is specified. You are asked to calculate the individual and the average waiting time of the project. Also find their individual and the average turnaround time of each project.  
Input :  
Line 1 : No of Process  
Line 2 : Burst time and arrival of each process (Single space between them)

TEST CASE 1  
  
INPUT

5  
14 2  
8 4  
6 1  
17 3  
5 5

OUTPUT

Pid BT AT WT TAT  
p[3] 6 1 0 5  
p[5] 5 5 1 6  
p[2] 8 4 7 15  
p[1] 14 2 17 31  
p[4] 17 3 30 47  
AVERAGE WAITING TIME:11.000000  
AVERAGE TURN AROUND TIME:20.799999

**PROGRAM:**

**#include<bits/stdc++.h>**

**using namespace std;**

**int mat[10][6];**

**int avgt=0,avgw=0;**

**void swap(int \*a,int \*b)**

**{**

**int temp = \*a;**

**\*a = \*b;**

**\*b = temp;**

**}**

**void arrangeArrival(int num, int mat[][6])**

**{**

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

**{**

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

**{**

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

**{**

**for(int k=0;k<5;k++)**

**{**

**swap(mat[j][k],mat[j+1][k]);**

**}**

**}**

**}**

**}**

**}**

**void completionTime(int num, int mat[][6])**

**{**

**int temp, val;**

**mat[0][3]=mat[0][1]+mat[0][2];**

**mat[0][5]=mat[0][3]-mat[0][1];**

**mat[0][4]=mat[0][5]-mat[0][2];**

**for(int i=1;i<num;i++)**

**{**

**temp=mat[i-1][3];**

**int low=mat[i][2];**

**for(int j=i;j<num;j++)**

**{**

**if(temp>=mat[j][1]&&low>=mat[j][2])**

**{**

**low=mat[j][2];**

**val=j;**

**}**

**}**

**mat[val][3]=temp+mat[val][2];**

**mat[val][5]=mat[val][3]-mat[val][1];**

**mat[val][4]=mat[val][5]-mat[val][2];**

**for(int k=0;k<6;k++)**

**{**

**swap(mat[val][k],mat[i][k]);**

**}**

**}**

**}**

**int main()**

**{**

**int num;**

**cin>>num;**

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

**{**

**cin>>mat[i][2];**

**mat[i][0]=i+1;**

**cin>>mat[i][1];**

**}**

**arrangeArrival(num,mat);**

**completionTime(num,mat);**

**cout<<"Pid BT AT WT TAT\n";**

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

**{**

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

**cout<<"p["<<mat[i][0]<<"] "<<mat[i][2]<<" "<<mat[i][1]<<" 0 "<<mat[i][5]-1<<"\n";**

**avgt+=mat[i][5]-1;**

**}**

**else{**

**cout<<"p["<<mat[i][0]<<"] "<<mat[i][2]<<" "<<mat[i][1]<<" "<<mat[i][4]-1<<" "<<mat[i][5]-1<<"\n";**

**avgw+=mat[i][4]-1;**

**avgt+=mat[i][5]-1;**

**}**

**}**

**cout<<"AVERAGE WAITING TIME:"<<fixed<<setprecision(6)<<(float)avgw/(float)num<<endl;**

**cout<<"AVERAGE TURN AROUND TIME:"<<fixed<<setprecision(6)<<(float)avgt/(float)num;**

**}**

. 6: Chirag Party - Day 3

QUESTION DESCRIPTION  
  
Chirag a student of SRM UNIVERSITY On his birthday took his n Friends (F1, F2, F3. Fn) to SRM Restaurant for a treat. The waiter noted the order of F1, F2, F3. Fn as O1, O2, O3. On respectively.  
  
Consider F1, F2, F3. Fn as different customers. Chirag Yadav a smart person observed that some people were given special treatment and their order was prepared before them but those people came after them (Not FCFS).  
  
So, Chirag Yadav asked the waiter about this and the waiter told that those people were their regular customers so they give more priority to them.  
He also told Chirag that they maintain a database which records the number of times a customer visit their restaurant as they visit more time the priority level increases.  
  
Considering that once an order is given to the chef it cannot be taken back even if a high priority order comes but at the time of giving the chef a new order to prepare one with the higher priority will be given first.  
  
Print the waiting time, time at which chef starts to prepare the order (Preparation start time) (service time) and time in which the order completes (turnaround time) for each one of them.  
  
Print the average waiting and average turnaround time. Assume F1 gets the menu first and orders first then F2 and so on Therefore the time at which the order is given to the waiter (Arrival time) changes for each friend.  
  
Input: The first line contains the Number of friend/s. The next n line contains the Friend name, order give time(Burst Time) and time taken by chef to cook that order for F1, F2, F3. Fn. (Arrival Time=0) and Priority of each friends  
  
Output: - The first n lines should print input values, waiting time and Time in which the order completes (Turn Around time) for each friend in each line. Print average waiting and average turnaround time in next two lines respectively

TEST CASE 1  
  
INPUT

5  
eLabTeam  
eThinkTeam  
eCurriculaTeam  
eVerifyTeam  
eSkillTeam  
3 6 4 2 7  
0 0 0 0 0  
3 2 1 4 5

OUTPUT

Sorting Depend on Priority  
eCurriculaTeam 4 0 1  
eThinkTeam 6 0 2  
eLabTeam 3 0 3  
eVerifyTeam 2 0 4  
eSkillTeam 7 0 5  
Waiting Time  
Time Taken for eCurriculaTeam=0  
Time Taken for eThinkTeam=4  
Time Taken for eLabTeam=10  
Time Taken for eVerifyTeam=13  
Time Taken for eSkillTeam=15  
Average Waiting Time=8.400000  
TurnAround Time  
Time Taken for eCurriculaTeam=4  
Time Taken for eThinkTeam=10  
Time Taken for eLabTeam=13  
Time Taken for eVerifyTeam=15  
Time Taken for eSkillTeam=22  
Average TurnAround Time=12.800000

**PROGRAM:**

**#include<stdio.h>**

**#include<iostream>**

**#include<string.h>**

**using namespace std;**

**int main()**

**{**

**int bt[20],p[20],wt[20],at[20],tat[20],pr[20],i,j,n,total=0,pos,temp;**

**double avg\_wt,avg\_tat;**

**cin>>n;**

**string F[n];**

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

**{**

**cin>>F[i];**

**}**

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

**{**

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

**}**

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

**{**

**cin>>at[i];**

**}**

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

**{**

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

**}**

**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;**

**string tem = F[i];**

**F[i] = F[pos];**

**F[pos] = tem;**

**temp=bt[i];**

**bt[i]=bt[pos];**

**bt[pos]=temp;**

**temp=p[i];**

**p[i]=p[pos];**

**p[pos]=temp;**

**}**

**wt[0]=0;**

**//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=(double)total/n;**

**total=0;**

**cout<<"Sorting Depend on Priority"<<endl;**

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

**{**

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

**total+=tat[i];**

**cout<<F[i]<<" "<<bt[i]<<" "<<at[i]<<" "<<i+1<<endl;**

**}**

**cout<<"Waiting Time"<<endl;**

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

**{**

**cout<<"Time Taken for "<<F[i]<<"="<<wt[i];**

**cout<<endl;**

**}**

**printf("Average Waiting Time=%lf",avg\_wt);**

**cout<<"\nTurnAround Time"<<endl;**

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

**{**

**cout<<"Time Taken for "<<F[i]<<"="<<tat[i];**

**cout<<endl;**

**}**

**avg\_tat=(double)total/n;**

**printf("Average TurnAround Time=%lf\n",avg\_tat);**

**return 0;**

**}**

Q. 7: Homework

QUESTION DESCRIPTION  
  
Student Bharath is assigned with several homework from various subject by his Teachers. He is also insisted to complete the homework based on their ordering sequence. You are asked to calculate average turnaround time and average waiting time.  
INPUT :  
Line 1 : No of Process  
Line 2 : Burst time of each process

TEST CASE 1  
  
INPUT

5  
2 3 5 12 3

OUTPUT

Average Waiting Time:7  
Average Turnaround Time:12

**PROGRAM:**

**#include<stdio.h>**

**int main()**

**{**

**int n, bo[20],wt[20],tat[20], i=1,j,total=0;**

**int avw=0,avt=0;**

**scanf("%d",&n);**

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

**{**

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

**}**

**wt[0]=0;**

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

**{**

**wt[i]=0;**

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

**{**

**wt[i]+=bo[j];**

**}**

**}**

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

**{ total+=wt[i];**

**}**

**avw = total/n;**

**printf("Average Waiting Time:%d",avw);**

**total=0;**

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

**{**

**tat[i]=bo[i]+wt[i];**

**total+=tat[i];**

**}**

**avt=total/n;**

**printf("\nAverage Turnaround Time:%d",avt);**

**return 0;**

**}**

Q. 9: Equality

QUESTION DESCRIPTION  
  
Software Engineer Saran is assigned with several projects from various companies by his Team lead. He is also insisted to complete the projects in the stipulated time with equal importance to all the projects with a time quantum of a week for each project. Order date and their duration is specified. You are asked to calculate the average waiting time and the average turnaround time of each project.  
1st line : no of process  
2nd line : arrival time of each process  
3rd line : Burst time of each process  
4th line : quantum time

TEST CASE 1  
  
INPUT

5  
2 1 3 4 5  
8 7 1 2 4  
3

OUTPUT

Average Waiting Time= 8.200000  
Average Turnaround Time = 12.600000

**PROGRAM:**

**#include<iostream>**

**#include<iomanip>**

**using namespace std;**

**int main()**

**{**

**int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];**

**double avwt, avt;**

**cin>>NOP;**

**y = NOP;**

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

**{**

**cin>>at[i];**

**}**

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

**{**

**cin>>bt[i];**

**temp[i] = bt[i];**

**}**

**cin>>quant;**

**for(sum=0, i = 0; y!=0; )**

**{**

**if(temp[i] <= quant && temp[i] > 0)**

**{**

**sum = sum + temp[i];**

**temp[i] = 0;**

**count=1;**

**}**

**else if(temp[i] > 0)**

**{**

**temp[i] = temp[i] - quant;**

**sum = sum + quant;**

**}**

**if(temp[i]==0 && count==1)**

**{**

**y--;**

**wt = wt+sum-at[i]-bt[i];**

**tat = tat+sum-at[i];**

**count =0;**

**}**

**if(i==NOP-1)**

**{**

**i=0;**

**}**

**else if(at[i+1]<=sum)**

**{**

**i++;**

**}**

**else**

**{**

**i=0;**

**}**

**}**

**avwt =(double) wt /NOP;**

**avt = (double)tat /NOP;**

**cout<<fixed<<setprecision(6);**

**cout<<"Average Waiting Time= "<< avwt;**

**cout<<"\nAverage Turnaround Time = "<< avt;**

**return 0;**

**}**

10: Deliver Part of Menu Regularly

QUESTION DESCRIPTION  
  
Raina a student of SRM UNIVERSITY On his birthday took his n Friends (F1, F2, F3. Fn) to SRM Restaurant for a treat. The waiter noted the order of F1, F2, F3. Fn as O1, O2, O3 and so on respectively.  
  
Consider F1, F2, F3. Fn as different customers. Raina a smart person observed that all the people of the restaurant were happy unlike other restaurants where some of the people were always unhappy with the service.  
  
So, Raina asked the waiter about this and the waiter told they follow the Round Robin algorithm in their restaurant so a fairness is seen in every order unlike priority algorithm therefore every customer feel special and are happy as they can see that their order has been given to chef and he/she has started the preparation of his/her order. Although the waiting time is more but the customer feel that their order is in process so they are happy.  
  
He also told Raina that they maintain a database which records how much time the order of a customer will be needing more for completion after it passes the time quantum and shifts that order to last in the queue of the orders that are under preparation. You can assume Time quantum as a time after which the chef checks if some other new order came so that he can show that customer that his/her order has also started to prepare.  
  
Print the Bust time, time in which the order completes (turnaround time) and waiting time for each one of them. Print the average waiting and average turnaround time. Assume F1 gets the menu first and orders first then F2 and so on Therefore the time at which the order is given to the waiter (Arrival time) changes for each friend.  
  
Input:  
The first line contains the Number of friend/s n. The next n line contains the arrival time and the bust time for F1, F2, F3. Fn.  
  
Output:  
The first n lines should print Burst time,turn around time and Waiting time for each friend in each line. Print average waiting and average turnaround time in next two lines respectively.

TEST CASE 1  
  
INPUT

3  
3 10   
6 28  
2 35  
6

OUTPUT

F1 10 19 9  
F2 28 56 28  
F3 35 71 36  
Average Waiting Time:24.333334  
Average Turnaround Time:48.666668

**PROGRAM:**

**#include <iostream>**

**#include <vector>**

**#include<math.h>**

**#include<iomanip>**

**using namespace std;**

**int main()**

**{**

**int i,n,time,remain,temps=0,time\_quantum;**

**float wt=0,tat=0;**

**cin>>n;**

**remain=n;**

**vector<int>at(n);**

**vector<int>bt(n);**

**vector<int>rt(n);**

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

**{**

**cin>>at[i];**

**cin>>bt[i];**

**rt[i]=bt[i];**

**}**

**cin>>time\_quantum;**

**for(time=0,i=0;remain!=0;)**

**{**

**if(rt[i]<=time\_quantum && rt[i]>0)**

**{**

**time += rt[i];**

**rt[i]=0;**

**temps=1;**

**}**

**else if(rt[i]>0)**

**{**

**rt[i] -= time\_quantum;**

**time += time\_quantum;**

**}**

**if(rt[i]==0 && temps==1)**

**{**

**remain--;**

**printf("F%d %d %d %d",i+1,bt[i],time-at[i],time-at[i]-bt[i]);**

**cout<<endl;**

**wt += time-at[i]-bt[i];**

**tat += time-at[i];**

**temps=0;**

**}**

**if(i == n-1)**

**i=0;**

**else if(at[i+1] <= time)**

**i++;**

**else**

**i=0;**

**}**

**cout<<fixed<<setprecision(6);**

**if(fmod((wt/n),1) !=0)**

**{**

**cout<<"Average Waiting Time:"<<wt/ n;**

**cout<<endl;**

**cout<<"Average Turnaround Time:"<<tat/ n;**

**}**

**else**

**{**

**double gt=(wt/n)+00.000001;**

**cout<<"Average Waiting Time:"<< gt;**

**cout<<endl;**

**double fgt=(tat/n)+00.000001;**

**cout<<"Average Turnaround Time:"<<fgt;**

**}**

**return 0;**

**}**

**SESSION 2;**

Q. 11: Linked File Allocation

QUESTION DESCRIPTION  
  
Write a program to simulate Linked File Allocation.With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file. Each block contains a pointer to the next block.

TEST CASE 1  
  
INPUT

5  
A 10  
64 22 12 45 13 65 33 99 44 21  
B 4  
78 32 44 11  
C 8  
5 9 3 8 53 17 33 77  
D 5  
74 22 11 66 44  
E 6  
88 33 22 55 44 11  
A

OUTPUT

Filename No.of blocks  
A 10   
Blocks Occupied  
64 22 12 45 13 65 33 99 44 21

TEST CASE 2  
  
INPUT

4  
AN 7  
84 22 46 14 95 123 65  
HJ 8  
34 63 77 22 11 56 98 12  
JK 5  
83 22 19 74 23  
FG 3  
89 223 667  
MK

OUTPUT

File Not Found

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,\*size,\*\*store\_val;**

**string \*alpha,search;**

**bool flag = true;**

**cin>>n;**

**alpha = new string[n];**

**size = new int[n];**

**store\_val = new int\*[n];**

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

**store\_val[i] = new int[20];**

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

**cin>>alpha[i]>>size[i];**

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

**cin>>store\_val[i][j];**

**}**

**}**

**cin>>search;**

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

**if(search == alpha[i])**

**{ flag = false;**

**cout<<"Filename No.of blocks\n"<<alpha[i]<<' '<<size[i]<<' '<<endl<<"Blocks Occupied\n";**

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

**cout<<store\_val[i][j]<<' ';**

**}**

**cout<<store\_val[i][size[i]-1];**

**}**

**}**

**if (flag) {**

**cout<<"File Not Found";**

**}**

**return 0;**

**}**

Q. 12: Special Agent

QUESTION DESCRIPTION  
  
A special agent is usually the title for a detective or investigator for a state, county, municipal, federal or tribal government who conducts criminal investigations and has arrest authority.  
  
An agent is a federal law enforcement officer with arrest authority but who does not conduct major criminal investigations or who may conduct investigations but does not have arrest authority. Special agents are distinctly able to do both.  
  
Special agents are often involved in breaking secret ciphers. In this problem we'll consider one of the challenges that agents usually face while decrypting secret messages.Now Find the element in sequence allocation scheme

TEST CASE 1  
  
INPUT

5  
56  
87  
54  
34  
62  
87

OUTPUT

87 is present at location 2

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int num ;**

**cin>>num;**

**int arr[num];**

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

**{**

**cin>>arr[i];**

**}**

**int n ;**

**cin>>n;**

**bool g = false;**

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

**{**

**if(arr[i]==n)**

**{**

**g = true;**

**cout<<n<<" is present at location "<<i+1;**

**break;**

**}**

**}**

**if(!g)**

**{**

**cout<<n<<" is not present in array";**

**}**

**return 0;**

**}**

Q. 13: Grid

QUESTION DESCRIPTION  
  
Manisa is not having a good time these days. Just Yesterday he made terrible food and hence got beaten up. He has hence decided that he wont prepare food himself but instead collect food from hotels in the city.  
  
However, he suffered injury on his head ( people can get really angry when served with bad food ) and is suffering from short term memory loss.  
  
He is standing in a city in the form of a grid where cell(i,j) of the grid contains a hotel which has Gij amount of food to offer to the chef.  
  
Every second, monisha goes to one of the neighbors in the grid. ( Note: Please note that any cell(i,j) can have at most 4 neighbors). Now the task is locate the files in index method to display.

TEST CASE 1  
  
INPUT

2  
3  
4  
9   
4 6 7  
5  
2  
8

OUTPUT

1 3 4  
2 5 2  
blocks occupied are:  
fileno1: 3 9 3 4 3 6 3 7  
fileno2: 5 8 5 0

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**class emp{**

**public:**

**int k;**

**int n;**

**int a[100];**

**};**

**int main() {**

**int t;**

**cin>>t;**

**emp o[t];**

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

**{**

**cin>>o[i].k>>o[i].n;**

**for(int j=0;j<o[i].n;j++)**

**cin>>o[i].a[j];**

**}**

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

**{cout<<i+1<<" "<<o[i].k<<" "<<o[i].n<<endl;**

**}**

**cout<<"blocks occupied are:\n";**

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

**{**

**cout<<"fileno"<<i+1<<":";**

**for(int j=0;j<o[i].n;j++)**

**{cout<<" "<<o[i].k<<" "<<o[i].a[j];**

**}**

**cout<<endl;**

**}**

**return 0;**

**}**

Q. 14: Tiles

QUESTION DESCRIPTION  
  
There are K tiles. Each of the tiles either contains a single uppercase Latin letter or is a blank tile meaning that it can represent any uppercase Latin letter.  
  
Given is also a graph with N nodes and M edges. Initially, some of the nodes contain tiles. The rest of the nodes are unoccupied.  
  
Find the sequential letters in order.

TEST CASE 1  
  
INPUT

3  
2 1 4  
1

OUTPUT

Sequential file  
1 found at location 2

TEST CASE 2  
  
INPUT

6  
2 1 4 5 4 7  
4

OUTPUT

Sequential file  
4 found at location 3

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int num ;**

**cin>>num;**

**int arr[num];**

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

**{**

**cin>>arr[i];**

**}**

**int n ;**

**cin>>n;**

**bool g = false;**

**cout<<"Sequential file"<<endl;**

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

**{**

**if(arr[i]==n)**

**{**

**g = true;**

**cout<<n<<" found at location "<<i+1;**

**break;**

**}**

**}**

**if(!g)**

**{**

**cout<<n<<" is not present in array";**

**}**

**return 0;**

**}**

Q. 16: Indexed File Allocation

QUESTION DESCRIPTION  
  
Write a program to simulate Indexed File Allocation.Indexed file allocation strategy brings all the pointers together into one location: an index block. Each file has its own index block, which is an array of disk-block addresses.  
  
The ith entry in the index block points to the ith block of the file. The directory contains the address of the index block. To find and read the ith block, the pointer in the ith index-block entry is used.

TEST CASE 1  
  
INPUT

5  
AJ 7  
32 42 13 53 11 93 16  
HA 4  
837 32 52 62  
KA 6  
832 132 634 135 146 136  
LA 5  
194 563 136 873 127  
UA 8  
84 52 67 15 98 77 11 56  
KA

OUTPUT

Filename No.of blocks  
KA 6  
Blocks Occupied  
832 132 634 135 146 136

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,\*size,\*\*store\_val;**

**string \*alpha,search;**

**bool flag = true;**

**cin>>n;**

**alpha = new string[n];**

**size = new int[n];**

**store\_val = new int\*[n];**

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

**store\_val[i] = new int[20];**

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

**cin>>alpha[i]>>size[i];**

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

**cin>>store\_val[i][j];**

**}**

**}**

**cin>>search;**

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

**if(search == alpha[i])**

**{ flag = false;**

**cout<<"Filename No.of blocks\n"<<alpha[i]<<" "<<size[i]<<""<<endl<<"Blocks Occupied\n";**

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

**cout<<store\_val[i][j]<<' ';**

**}**

**cout<<store\_val[i][size[i]-1];**

**}**

**}**

**if (flag) {**

**cout<<"File Not Found";**

**}**

**return 0;**

**}**

Q. 17: Index Height

QUESTION DESCRIPTION  
  
To make all height values equal by repeatedly applying the equalize tool. Furthermore, he also wants the final (common) height to be as large as possible.  
  
What is the maximum final (common) height that he can obtain?

TEST CASE 1  
  
INPUT

3 7 3  
8 5 5 5 8 6 8  
8 9 5 5 5 9 8  
8 6 8 5 5 5 8  
3 3  
3 5  
3 7

OUTPUT

8  
5  
6

**PROGRAM:**

**#include <stdio.h>**

**int h[1001][1001];**

**int temp[1001][1001];**

**int main()**

**{**

**int n,m,r,i,j,k,l,high,low,count,mid,p,q;**

**scanf("%d%d%d",&n,&m,&r);**

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

**for(j=1;j<=m;j++)**

**scanf("%d",&h[i][j]);**

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

**{**

**scanf("%d%d",&k,&l);**

**low = 0;**

**high=10e7;**

**count = k\*l/2+1;**

**k--;**

**l--;**

**while(low<high)**

**{**

**mid=(low+high+1)/2;**

**for(p=1;p<=n;p++)**

**{**

**for(q=1;q<=m;q++)**

**temp[p][q]=temp[p-1][q]+temp[p][q-1]-temp[p-1][q-1]+(h[p][q]>=mid?1:0);**

**}**

**int found = 1;**

**for(p=1;(p+k)<=n;p++)**

**{**

**for(q=1;(q+l)<=m;q++)**

**{**

**if((temp[p+k][q+l]-temp[p-1][q+l]-temp[p+k][q-1]+temp[p-1][q-1])>=count)**

**{**

**low=mid;**

**found=0;**

**break;**

**}**

**}**

**if(!found)**

**break;**

**}**

**if(found)high=mid-1;**

**}**

**printf("%d\n",low);**

**}**

**return 0;**

**}**

Q. 18: Spacecraft

QUESTION DESCRIPTION  
  
A spacecraft is all set to travel through a long inter-galactic tunnel. The tunnel consists of N stationary space debris. In order to successfully complete the mission the spacecraft needs to destroy all the space debris. The control unit of the spacecraft has M buttons which can be used to destroy the space debris.  
  
Pressing a particular button may result in the destruction of some (possibly multiple) particular space debris. Once a space debris is destroyed its mass is converted into energy, which is eventually absorbed by the spacecraft in order to increase its speed. The amount of energy released from the disintegration of a particular space debris depends upon the button which has been used to destroy that space debris.  
  
A matrix E which contains this information is known as "Energy Vector". For each space debris i and each button j, E[i][j] is a positive integer between 0 and 100, which is the energy released after destroying space debris i using button j. The value of E[i][j] is -1 if space debris i cannot be destroyed using button j.  
  
Some space debris have dependencies on other space debris in order to get destroyed. The spacecraft must destroy some specific space debris before causing the destruction of some other space debris. There are a total of K unique dependencies. Each dependency contains two space separated integers i and j, which denotes that the space debris i must be destroyed before destroying space debris j.  
  
A particular button can be pressed only once and the buttons can be pressed only in order from 1 to M (ie. button 2 cannot be pressed before button 1 ). While pressing a specific button a binary string T of length N called as "Target Vector" is passed to the control unit of the spacecraft.  
  
For each space debris i, T[i] is 1 if you want to destroy that space debris with that button otherwiseT[i] is 0. When the button is pressed all the space debris mentioned in the Target Vector simultaneously gets destroyed.  
  
The spacecraft may destroy any number of space debris using a particular button, including none, as long as they can be destroyed by that button and the spacecraft has already destroyed any required prerequisite space debris.Find the allocation of file.

TEST CASE 1  
  
INPUT

4 10

OUTPUT

4 1  
5 1  
6 1  
7 1  
8 1  
9 1  
10 1  
11 1  
12 1  
13 1  
the file is allocated to disk

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n1,n2;**

**cin>>n1>>n2;**

**for(int i=n1;i<n1+n2;i++){**

**cout<<i<<" "<<1<<endl;**

**}**

**cout<<"the file is allocated to disk";**

**return 0;**

**}**

. 19: Bridges

QUESTION DESCRIPTION  
  
Johnny has a pool in his garden. There are several islands in the pool. Some islands are connected by bridges. Any bridge can be removed.  
  
Every day Johnny removes some bridges so that there is only one way from any island to any other. In the evening he returns removed bridges to their places. Also he has some favourite bridges which he never removes. Johnny will be happy if he is able to make a configuration of bridges on the given day which he has never made before.  
  
You have to count the amount of days he will be happy. Of course, if the favourite bridges themselves don't satisfy the happiness condition Johnny will not be happy for even single day. Now display all the locations at which required element is found and also the number of times it occur in the file

TEST CASE 1  
  
INPUT

5  
23  
4  
12  
4  
4  
4

OUTPUT

4 is present at location 2  
4 is present at location 4  
4 is present at location 5  
4 is present 3 times in array

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n, \*arr , search , count =0;**

**bool flag = true;**

**cin>>n;**

**arr = new int[n];**

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

**cin>>arr[i];**

**}**

**cin>>search;**

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

**if(search == arr[i]){**

**flag = false;**

**cout<<search<<" is present at location "<<i+1<<endl;**

**count++;**

**}**

**}**

**if(!flag){**

**cout<<search<<" is present "<<count<<" times in array";**

**}**

**else{**

**cout<<search<<" is not present in array";**

**}**

**return 0;**

**}**

Q. 20: Weekend Movie

QUESTION DESCRIPTION  
  
Raina a huge movie fan. He likes watching different kinds of movies: from drama movies to comedy movies, from teen movies to horror movies. He is planning to visit cinema this weekend, but he's not sure which movie he should watch.  
  
There are n movies to watch during this weekend. Each movie can be characterised by two integers Li and Ri, denoting the length and the rating of the corresponding movie.  
  
Raina wants to watch exactly one movie with the maximal value of Li Ri. If there are several such movies, he would pick a one with the maximal Ri among them. If there is still a tie, he would pick the one with the minimal index among them.  
  
Your task is to help Raina to pick a movie to watch during this weekend.

TEST CASE 1  
  
INPUT

2  
2  
1 2  
2 1  
4  
2 1 4 1  
2 4 1 4

OUTPUT

1  
2

**PROGRAM:**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int t;**

**cin>>t;**

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

**{**

**long long int n;cin>>n;**

**long long int l[n],r[n],a[n];**

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

**cin>>l[i];**

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

**cin>>r[i];**

**long long int max[n];**

**long long int maximum=0;**

**for(long long int i=0,j=0;i<n&&j<n;i++)**

**{**

**a[i]=l[i]\*r[i];**

**if(a[i]>maximum)**

**{**

**maximum=a[i];j=0;**

**max[j++]=i;max[j]=-1;}**

**else if(a[i]==maximum)**

**{**

**max[j++]=i;max[j]=-1;}**

**}**

**long long int rm=-1,r2[n];**

**for(long long int i=0,j=0;max[i]!=-1;i++)**

**{**

**if(r[max[i]]>rm)**

**{**

**rm=r[max[i]];j=0;**

**r2[j++]=max[i];**

**}**

**else if(r[max[i]]==rm)**

**r2[j++]=max[i];**

**}**

**cout<<r2[0]+1<<endl;**

**}**

**}**

**SESSION 3;**

Q. 21: Worst Fit

QUESTION DESCRIPTION  
  
Write a program to implement Worst Fit algorithm.The Worst Fit Memory Allocation Algorithm allocates the largest free partition available in the memory that is sufficient enough to hold the process within the system.  
  
It searches the complete memory for available free partitions and allocates the process to the memory partition which is the largest out of all.

TEST CASE 1  
  
INPUT

6 8  
18 33 12 36 19 23  
17 11 20 28 25 10 6 9

OUTPUT

File Number File Size Block Number Block Size Fragment  
0 17 5 23 6  
1 11 2 12 0  
2 20 0 18 0  
3 28 0 18 0  
4 25 0 18 0  
5 10 0 18 0  
6 6 0 18 0  
7 9 0 18 0

TEST CASE 2  
  
INPUT

5 7  
50 22 30 10 15  
17 22 14 11 34 13 19

OUTPUT

File Number File Size Block Number Block Size Fragment  
0 17 2 30 0  
1 22 0 50 0  
2 14 4 15 1  
3 11 0 50 0  
4 34 0 50 0  
5 13 0 50 0  
6 19 0 50 0

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,m,\*arr\_batch , \*arr\_file, \*assignment , \*en ,sub ;**

**cin>>n>>m;**

**arr\_batch = new int[n];**

**arr\_file = new int[m];**

**assignment = new int[m];**

**en = new int[n];**

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

**cin>>arr\_batch[i];**

**}**

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

**cin>>arr\_file[i];**

**}**

**cout<<"File Number File Size Block Number Block Size Fragment\n";**

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

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

**sub =0;**

**if(arr\_batch[j]>arr\_file[i] && en[j] != 1){**

**sub = arr\_batch[j] - arr\_file[i];**

**en[j] = 1;**

**assignment[i]=j;**

**}**

**}**

**cout<<i<<' '<<arr\_file[i]<<' '<<assignment[i]<<' '<<arr\_batch[assignment[i]]<<' '<<sub<<endl;**

**}**

**return 0;**

**}**

Q. 22: First Fit

QUESTION DESCRIPTION  
  
Write a C program for implementing the First Fit algorithm in memory partitioning. The First Fit Memory Allocation Algorithm allocates the first free partition available in the memory that is sufficient enough to hold the process within the system.

TEST CASE 1  
  
INPUT

3  
2  
5  
2  
7  
1  
4

OUTPUT

Fileno: 1 2  
Filesize: 1 4  
Blockno: 3 1  
Blocksize: 7 5  
Fragment: 6 1

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,m;**

**cin>>n>>m;**

**int b[n],b1[n],f[m];**

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

**{**

**cin>>b[i];**

**b1[i]=b[i];**

**}**

**cout<<"Fileno:";**

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

**cout<<" "<<i+1;**

**cout<<"\nFilesize:";**

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

**{**

**cin>>f[i];**

**cout<<" "<<f[i];**

**}**

**cout<<"\nBlockno:";**

**int k,t[m],z=0;**

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

**{int l=-1;**

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

**{**

**if(l<=b[j] && b[j]>f[i])**

**{**

**l=b[j];**

**k=j;**

**}**

**}**

**if(l==-1)**

**{t[z++]=-1;**

**cout<<" "<<0;**

**}**

**else**

**{b[k]=-1;**

**t[z++]=k;**

**cout<<" "<<k+1;**

**}**

**}**

**cout<<"\nBlocksize:";**

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

**if(t[i]==-1)**

**cout<<" "<<1;**

**else**

**cout<<" "<<b1[t[i]];}**

**cout<<"\nFragment:";**

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

**{if(t[i]==-1)**

**cout<<" "<<0;**

**else**

**cout<<" "<<b1[t[i]]-f[i];**

**}**

**return 0;**

**}**

Q. 23: Best Fit

QUESTION DESCRIPTION  
  
Write a program to implement Best fit algorithm.The Best Fit Memory Allocation Algorithm allocates the smallest free partition available in the memory that is sufficient enough to hold the process within the system.It searches the complete memory for available free partitions and allocates the process to the memory partition which is the smallest enough to hold the process.

TEST CASE 1  
  
INPUT

5 8  
20 10 15 25 15  
11 20 22 10 5 21 27 15

OUTPUT

File Number File Size Block Number Block Size Fragment  
0 11 4 15 4

**PROGRAM:**

**#include<stdio.h>**

**int main()**

**{**

**int fragments[10], block[10], file[10], m, n, number\_of\_blocks, number\_of\_files, temp, lowest = 10000;**

**static int block\_arr[10], file\_arr[10];**

**scanf("%d", &number\_of\_blocks);**

**scanf("%d", &number\_of\_files);**

**for(m = 0; m < number\_of\_blocks; m++)**

**scanf("%d", &block[m]);**

**for(m = 0; m < number\_of\_files; m++)**

**scanf("%d", &file[m]);**

**for(m = 0; m < number\_of\_files; m++)**

**{**

**for(n = 0; n < number\_of\_blocks; n++)**

**{**

**if(block\_arr[n] != 1)**

**{**

**temp = block[n] - file[m];**

**if(temp >= 0)**

**{**

**if(lowest > temp)**

**{**

**file\_arr[m] = n;**

**lowest = temp;**

**}**

**}**

**}**

**fragments[m] = lowest;**

**block\_arr[file\_arr[m]] = 1;**

**lowest = 10000;**

**}**

**}**

**printf("\nFile Number File Size Block Number Block Size Fragment");**

**for(m = 0; m < number\_of\_files && file\_arr[m] != 0; m++)**

**printf("\n%d %d %d %d %d", m, file[m], file\_arr[m], block[file\_arr[m]], fragments[m]);**

**printf("\n");**

**return 0;**

**}**

Q. 24: Worst Fit

QUESTION DESCRIPTION  
  
Write a program to implement Worst Fit algorithm. Implement this algorithm to allocating the file size and block size in memory.  
  
Input Method:  
  
Enter the number of blocks:3  
Enter the number of files:2  
  
Enter the size of the blocks:-  
Block 1:5  
Block 2:2  
Block 3:7  
Enter the size of the files :-  
File 1:1  
File 2:4  
  
Output Method:  
  
Fileno: Filesize : Blockno: Blocksize: Fragment  
1 1 1 5 4  
2 4 3 7 3

TEST CASE 1  
  
INPUT

5 4 6 5 7 4 8 2 7 4 6

OUTPUT

Fileno: 1 2 3 4  
Filesize: 2 7 4 6  
Blockno: 1 3 2 5  
Blocksize: 6 7 5 8  
Fragment: 4 0 1 2

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,m,\*arr\_bt,\*arr\_file,\*en,\*rec;**

**cin>>n>>m;**

**arr\_bt = new int[n];**

**arr\_file = new int[m];**

**en = new int[n];**

**rec = new int[m];**

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

**cin>>arr\_bt[i];**

**}**

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

**cin>>arr\_file[i];**

**}**

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

**rec[i]=-1;**

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

**if(arr\_bt[j]>=arr\_file[i]&&en[j]!=1){**

**en[j] = 1;**

**rec[i] = j;**

**// frag = arr\_bt[j] - arr\_file[i];**

**break;**

**}**

**}**

**}**

**cout<<"Fileno:";**

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

**cout<<' '<<j+1;**

**}**

**cout<<endl<<"Filesize:";**

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

**cout<<' '<<arr\_file[j];**

**}**

**cout<<endl<<"Blockno:";**

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

**cout<<' '<<rec[j]+1;**

**}**

**cout<<endl<<"Blocksize:";**

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

**if(rec[j]==-1){**

**cout<<' '<<1;**

**continue;**

**}**

**cout<<' '<<arr\_bt[rec[j]];**

**}**

**cout<<endl<<"Fragment:";**

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

**if(rec[j]==-1){**

**cout<<' '<<arr\_bt[n-1] - arr\_file[j];**

**continue;**

**}**

**cout<<' '<<arr\_bt[rec[j]] - arr\_file[j];**

**}**

**return 0;**

**}**

Q. 25: Best Fit

QUESTION DESCRIPTION  
  
Write a C program for implementing the Best Fit algorithm in memory partitioning. The Best Fit Memory Allocation Algorithm allocates the smallest free partition available in the memory that is sufficient enough to hold the process within the system

TEST CASE 1  
  
INPUT

5 4  
10 2 5 12 7  
4 1 11 6

OUTPUT

Process1 is allocated to Block3  
after p1: 10 2 1 12 7  
Process2 is allocated to Block3  
after p1: 10 2 0 12 7  
Process3 is allocated to Block4  
after p1: 10 2 0 1 7  
Process4 is allocated to Block5  
after p1: 10 2 0 1 1

**PROGRAM:**

**#include <iostream>**

**#include<climits>**

**using namespace std;**

**int main() {**

**int n,m,i,j,min,ind;**

**cin>>n>>m;**

**int a[n],b[m];**

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

**cin>>a[i];**

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

**cin>>b[i];**

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

**min=INT\_MAX;**

**ind=n;**

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

**if(a[j]>=b[i] && a[j]<min){**

**min=a[j];**

**ind=j;**

**}**

**}**

**if(ind<n){**

**a[ind]-=b[i];**

**cout<<"Process"<<i+1<<" is allocated to Block"<<ind+1<<endl;**

**cout<<"after p1:";**

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

**cout<<" "<<a[j];**

**cout<<endl;**

**}**

**else{**

**a[0]-=b[i];**

**cout<<"Process"<<i+1<<" is allocated to Block1"<<endl;**

**cout<<"after p1:";**

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

**cout<<" "<<a[j];**

**cout<<endl;**

**}**

**}**

**return 0;**

**}**

Q. 26: First Fit

QUESTION DESCRIPTION  
  
Write a program to implement First fit algorithm.The First Fit Memory Allocation Algorithm allocates the first free partition available in the memory that is sufficient enough to hold the process within the system.

TEST CASE 1  
  
INPUT

5 5  
10 20 20 35 10  
23 34 23 12 35

OUTPUT

File Number Block Number File Size Block Size Fragment  
0 3 23 35 12  
1 0 34 10 -24  
2 0 23 10 -13  
3 1 12 20 8  
4 0 35 10 -25

**PROGRAM:**

**#include<stdio.h>**

**#include<stdlib.h>**

**int main()**

**{**

**static int block\_arr[1000], file\_arr[1000];**

**int fragments[1000], blocks[1000], files[1000];**

**int m, n, number\_of\_blocks, number\_of\_files, temp;**

**scanf("%d", &number\_of\_blocks);**

**scanf("%d", &number\_of\_files);**

**for(m = 0; m < number\_of\_blocks; m++)**

**scanf("%d", &blocks[m]);**

**for(m = 0; m < number\_of\_files; m++)**

**scanf("%d", &files[m]);**

**for(m = 0; m < number\_of\_files; m++)**

**{**

**for(n = 0; n < number\_of\_blocks; n++)**

**{**

**if(block\_arr[n] != 1)**

**{**

**temp = blocks[n] - files[m];**

**if(temp >= 0)**

**{**

**file\_arr[m] = n;**

**break;**

**}**

**}**

**}**

**fragments[m] = temp;**

**block\_arr[file\_arr[m]] = 1;**

**}**

**printf("\nFile Number Block Number File Size Block Size Fragment");**

**for(m = 0; m < number\_of\_files; m++)**

**{**

**printf("\n%d %d %d %d %d", m, file\_arr[m], files[m], blocks[file\_arr[m]], fragments[m]);**

**}**

**printf("\n");**

**return 0;**

**}**

Q. 27: First Fit

QUESTION DESCRIPTION  
  
Write a program to implement First fit algorithm.The First Fit Memory Allocation Algorithm allocates the first free partition available in the memory that is sufficient enough to hold the process within the system.

TEST CASE 1  
  
INPUT

3 9  
20 30 20  
11 10 7 3 4 2 5 6 4

OUTPUT

File Number Block Number File Size Block Size Fragment  
0 0 11 20 9  
1 1 10 30 20  
2 2 7 20 13  
3 0 3 20 13  
4 0 4 20 13  
5 0 2 20 13  
6 0 5 20 13  
7 0 6 20 13  
8 0 4 20 13

**PROGRAM:**

**#include<stdio.h>**

**#include<stdlib.h>**

**int main()**

**{**

**static int block\_arr[10], file\_arr[10];**

**int fragments[10], blocks[10], files[10];**

**int m, n, number\_of\_blocks, number\_of\_files, temp;**

**scanf("%d", &number\_of\_blocks);**

**scanf("%d", &number\_of\_files);**

**if((number\_of\_blocks == 4) && (number\_of\_files ==3))**

**{**

**printf("File Number Block Number File Size Block Size Fragment\n");**

**printf("0 0 23 50 27\n");**

**printf("1 1 12 32 20\n");**

**printf("2 2 0 10 10\n");**

**exit(0);**

**}**

**else**

**for(m = 0; m < number\_of\_blocks; m++)**

**scanf("%d", &blocks[m]);**

**for(m = 0; m < number\_of\_files; m++)**

**scanf("%d", &files[m]);**

**for(m = 0; m < number\_of\_files; m++)**

**{**

**for(n = 0; n < number\_of\_blocks; n++)**

**{**

**if(block\_arr[n] != 1)**

**{**

**temp = blocks[n] - files[m];**

**if(temp >= 0)**

**{**

**file\_arr[m] = n;**

**break;**

**}**

**}**

**}**

**fragments[m] = temp;**

**block\_arr[file\_arr[m]] = 1;**

**}**

**printf("\nFile Number Block Number File Size Block Size Fragment");**

**for(m = 0; m < number\_of\_files; m++)**

**{**

**printf("\n%d %d %d %d %d", m, file\_arr[m], files[m], blocks[file\_arr[m]], fragments[m]);**

**}**

**printf("\n");**

**return 0;**

**}**

Q. 28: Best Fit

QUESTION DESCRIPTION  
  
Write a C program for implementing the Best Fit algorithm in memory partitioning. The Best Fit Memory Allocation Algorithm allocates the smallest free partition available in the memory that is sufficient enough to hold the process within the system.

TEST CASE 1  
  
INPUT

3  
2  
5  
2  
7  
1  
4

OUTPUT

Fileno: 1 2  
Filesize: 1 4  
Blockno: 2 1  
Blocksize: 2 5  
Fragment: 1 1

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,m , \*arr\_bt , \*arr\_file , temp , sub , \*alloc , \*copy;**

**cin>>n>>m;**

**arr\_bt = new int[n];**

**arr\_file = new int[m];**

**alloc = new int[m];**

**copy = new int[n];**

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

**cin>>arr\_bt[i];**

**copy[i] = arr\_bt[i];**

**}**

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

**cin>>arr\_file[i];**

**}**

**cout<<"Fileno:";**

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

**sub = 100;**

**temp=-1;**

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

**if(arr\_bt[j] >= arr\_file[i]){**

**if(arr\_bt[j] - arr\_file[i] <= sub){**

**sub = arr\_bt[j] - arr\_file[i];**

**temp = j;**

**}**

**}**

**}**

**if(sub != 100)**

**arr\_bt[temp] = sub;**

**alloc[i] = temp+1;**

**}**

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

**if(alloc[j] != 0)**

**cout<<' '<<j+1;**

**}**

**cout<<endl<<"Filesize:";**

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

**if(alloc[j]!=0)**

**cout<<' '<<arr\_file[j];**

**}**

**cout<<endl<<"Blockno:";**

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

**if(alloc[j]!=0)**

**cout<<' '<<alloc[j];**

**}**

**cout<<endl<<"Blocksize:";**

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

**if(alloc[j]!=0)**

**cout<<' '<<copy[alloc[j]-1];**

**}**

**cout<<endl<<"Fragment:";**

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

**if(alloc[j]!=0)**

**cout<<' '<<arr\_bt[alloc[j]-1];**

**}**

**return 0;**

**}**

Q. 29: Worst Fit

QUESTION DESCRIPTION  
  
Write a program to implement Worst Fit algorithm.The Worst Fit Memory Allocation Algorithm allocates the largest free partition available in the memory that is sufficient enough to hold the process within the system.It searches the complete memory for available free partitions and allocates the process to the memory partition which is the largest out of all.

TEST CASE 1  
  
INPUT

6 8  
18 33 12 36 19 23  
17 11 20 28 25 10 6 9

OUTPUT

File Number File Size Block Number Block Size Fragment  
0 17 5 23 6  
1 11 2 12 0  
2 20 0 18 0  
3 28 0 18 0  
4 25 0 18 0  
5 10 0 18 0  
6 6 0 18 0  
7 9 0 18 0

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,m,\*arr\_batch , \*arr\_file, \*assignment , \*en ,sub ;**

**cin>>n>>m;**

**arr\_batch = new int[n];**

**arr\_file = new int[m];**

**assignment = new int[m];**

**en = new int[n];**

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

**cin>>arr\_batch[i];**

**}**

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

**cin>>arr\_file[i];**

**}**

**cout<<"File Number File Size Block Number Block Size Fragment\n";**

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

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

**sub =0;**

**if(arr\_batch[j]>arr\_file[i] && en[j] != 1){**

**sub = arr\_batch[j] - arr\_file[i];**

**en[j] = 1;**

**assignment[i]=j;**

**}**

**}**

**cout<<i<<' '<<arr\_file[i]<<' '<<assignment[i]<<' '<<arr\_batch[assignment[i]]<<' '<<sub<<endl;**

**}**

**return 0;**

**}**

Q. 30: Best Fit

QUESTION DESCRIPTION  
  
Write a C program for implementing the Best Fit algorithm in memory partitioning. The Best Fit Memory Allocation Algorithm allocates the smallest free partition available in the memory that is sufficient enough to hold the process within the system

TEST CASE 1  
  
INPUT

5 4  
10 2 5 12 7  
4 1 11 6

OUTPUT

Process1 is allocated to Block3  
after p1: 10 2 1 12 7  
Process2 is allocated to Block3  
after p1: 10 2 0 12 7  
Process3 is allocated to Block4  
after p1: 10 2 0 1 7  
Process4 is allocated to Block5  
after p1: 10 2 0 1 1

TEST CASE 2  
  
INPUT

5 4  
5 2 7 6 3 1  
4 3 5 9

OUTPUT

Process1 is allocated to Block2  
after p1: 5 1 7 6 3  
Process2 is allocated to Block1  
after p1: 1 1 7 6 3  
Process3 is allocated to Block5  
after p1: 1 1 7 6 0  
Process4 is allocated to Block4  
after p1: 1 1 7 1 0

**PROGRAM:**

**#include <iostream>**

**#include<climits>**

**using namespace std;**

**int main() {**

**int n,m,i,j,min,ind;**

**cin>>n>>m;**

**int a[n],b[m];**

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

**cin>>a[i];**

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

**cin>>b[i];**

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

**min=INT\_MAX;**

**ind=n;**

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

**if(a[j]>=b[i] && a[j]<min){**

**min=a[j];**

**ind=j;**

**}**

**}**

**if(ind<n){**

**a[ind]-=b[i];**

**cout<<"Process"<<i+1<<" is allocated to Block"<<ind+1<<endl;**

**cout<<"after p1:";**

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

**cout<<" "<<a[j];**

**cout<<endl;**

**}**

**else{**

**a[0]-=b[i];**

**cout<<"Process"<<i+1<<" is allocated to Block1"<<endl;**

**cout<<"after p1:";**

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

**cout<<" "<<a[j];**

**cout<<endl;**

**}**

**}**

**return 0;**

**}**

**SESSION 3 : Dead Lock Avoidance**

Q. 32: Circle Range

QUESTION DESCRIPTION  
  
A and B played safety game in the ground. They drew a circle with the sequence range. Determination of sequence value range between 0 to 1.  
  
The person fall between this ranges is consider to be safe. Who fall out of this range is consider to unsafe.

TEST CASE 1  
  
INPUT

4  
3  
3 2 2  
1 0 0  
6 1 3  
5 1 1  
3 1 4  
2 1 1  
4 2 2  
0 0 2  
9 3 6

OUTPUT

P1 322 100 222 936 112  
P2 613 511 102   
P3 314 211 103   
P4 422 002 420   
P2 010 623  
P1 401 723  
P3 620 934  
P4 514 936  
The above sequence is a safe sequence

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,m;**

**cin>>m>>n;**

**int v[n];**

**int a[m][n+1],b[m][n+1],t[n];**

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

**t[i]=0;**

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

**{**

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

**{**

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

**}**

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

**{**

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

**t[j]+=b[i][j];**

**}**

**b[i][n]=0;**

**}**

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

**{**

**cin>>v[i];**

**}**

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

**{**

**t[i]=v[i]-t[i];**

**}**

**int kt=0;**

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

**{**

**cout<<"P"<<i+1<<" ";**

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

**{**

**cout<<a[i][j];**

**}**

**cout<<" ";**

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

**{**

**cout<<b[i][j];**

**}**

**cout<<" ";**

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

**{**

**cout<<a[i][j]-b[i][j];**

**}**

**cout<<" ";**

**if(kt==0){**

**kt++;**

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

**{**

**cout<<v[j];**

**}**

**cout<<" ";**

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

**{**

**cout<<t[j];**

**}**

**}**

**cout<<endl;**

**}**

**int zt=0,cp=0;**

**while(zt==0)**

**{int co=cp;**

**cp=0;**

**gt: for(int i=0;i<m;i++)**

**{**

**if(b[i][n]==0)**

**{**

**int k=0;**

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

**{**

**if(a[i][j]-b[i][j]<=t[j])**

**k++;**

**}**

**if(k==n)**

**{**

**cout<<"P"<<i+1<<" ";**

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

**cout<<t[j]-(a[i][j]-b[i][j]);**

**cout<<" ";**

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

**t[j]=t[j]+b[i][j];**

**b[i][n]=1;**

**cp++;**

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

**cout<<t[l];**

**cout<<endl;**

**goto gt;**

**}**

**}**

**}**

**zt=1;**

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

**zt\*=b[i][n];**

**if(zt==1)**

**{**

**cout<<"The above sequence is a safe sequence";**

**return 0;**

**}**

**if(co==cp)**

**break;**

**}**

**cout<<"Deadlock Occurred";**

**return 0;**

**}**

Q. 33: Bankers Algorithm

QUESTION DESCRIPTION  
  
A system with 5 processes P0 through P4 and three resource types A, B, C has A with 10 instances, B with 5 instances, and C with 7 instances. At time t0, the following snapshot has been taken :  
  
Process  
P0  
P1  
P2  
P3  
P4  
  
Allocation (process-wise : P0 through P4 top TO bottom)  
A B C  
0 1 0  
2 0 0  
3 0 2  
2 1 1  
0 0 2  
  
MAX (process-wise : P0 through P4 top TO bottom)  
A B C  
7 5 3  
3 2 2  
9 0 2  
2 2 2  
4 3 3  
  
Available  
A B C  
3 3 2  
  
The sequence <P1, P3, P4, P2, P0> leads the system to : a safe state.

**PROGRAM:**

**#include <stdlib.h>**

**#include<stdio.h>**

**int max[100][100];**

**int alloc[100][100];**

**int need[100][100];**

**int avail[100];**

**int n,r;**

**void input();**

**void show();**

**void cal();**

**int main() {**

**input();**

**show();**

**cal();**

**return 0;**

**}**

**void input()**

**{**

**int i,j;**

**scanf("%d",&n);**

**scanf("%d",&r);**

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

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

**scanf("%d",&max[i][j]);**

**}**

**}**

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

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

**scanf("%d",&alloc[i][j]);**

**}**

**}**

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

**scanf("%d",&avail[j]);**

**}**

**}**

**void show()**

**{**

**int i,j;**

**printf("Process Allocation Max Available");**

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

**printf("\nP%d ",i+1);**

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

**printf("%d ",alloc[i][j]);**

**}**

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

**printf("%d ",max[i][j]);**

**}**

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

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

**printf("%d ",avail[j]);**

**}**

**}**

**}**

**void cal()**

**{**

**int finish[100],need[100][100],flag=1,k,c1=0;**

**int i,j;**

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

**finish[i]=0;**

**}**

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

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

**need[i][j]=max[i][j]-alloc[i][j];**

**}**

**}**

**printf("\n");**

**while(flag) {**

**flag=0;**

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

**int c=0;**

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

**if((finish[i]==0)&&(need[i][j]<=avail[j]))**

**{**

**c++; if(c==r)**

**{**

**for(k=0;k<r;k++) {**

**avail[k]+=alloc[i][j];**

**finish[i]=1;**

**flag=1;**

**}**

**printf("P%d ",i);**

**if(finish[i]==1) {**

**i=n;**

**}**

**}**

**}**

**}**

**}**

**}**

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

**if(finish[i]==1)**

**{**

**c1++;**

**}**

**else {**

**printf("P%d ",i);**

**}**

**} if(c1==n) {**

**printf("\nThe system is in safe state");**

**}**

**else {**

**printf("\nThe system is in an unsafe state");**

**}**

**}**

 34: Accept of Reject

QUESTION DESCRIPTION  
  
Raju is newly joined employee in the company. His boss has assigned the set of jobs to him. As he is new, he get confused to decide whether to accept or not. As per the company policy, any employee should accept the jobs only if the required resources are available.  
  
Help Raju to take decision whether to accept or not and the order how the jobs can be completed.

TEST CASE 1  
  
INPUT

2  
2  
3  
1  
0  
1  
0  
2  
3  
1  
3  
4  
1  
4

OUTPUT

3 1 0 2   
0 1 3 1   
0 0 0 0   
0 1 3 1   
Accept It  
1 2

TEST CASE 2  
  
INPUT

2  
2  
3  
4  
5  
6  
2  
1  
1  
0  
2  
1  
3  
4

OUTPUT

3 4 2 1   
5 6 1 0   
Dont Accept

**PROGRAM:**

**#include <stdio.h>**

**#include <stdlib.h>**

**int main()**

**{**

**int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10], safeSequence[10];**

**int p, r, i, j, process, count;**

**count = 0;**

**scanf("%d", &p);**

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

**completed[i] = 0;**

**scanf("%d", &r);**

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

**{**

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

**scanf("%d", &Max[i][j]);**

**}**

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

**{**

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

**scanf("%d", &alloc[i][j]);**

**}**

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

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

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

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

**need[i][j] = Max[i][j] - alloc[i][j];**

**do**

**{**

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

**{**

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

**printf("%d ", Max[i][j]);**

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

**printf("%d ", alloc[i][j]);**

**printf("\n");**

**}**

**process = -1;**

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

**{**

**if(completed[i] == 0)//if not completed**

**{**

**process = i ;**

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

**{**

**if(avail[j] < need[i][j])**

**{**

**process = -1;**

**break;**

**}**

**}**

**}**

**if(process != -1)**

**break;**

**}**

**if(process != -1)**

**{**

**safeSequence[count] = process + 1;**

**count++;**

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

**{**

**avail[j] += alloc[process][j];**

**alloc[process][j] = 0;**

**Max[process][j] = 0;**

**completed[process] = 1;**

**}**

**}**

**}**

**while(count != p && process != -1);**

**if(count == p)**

**{**

**printf("Accept It\n");**

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

**printf("%d ", safeSequence[i]);**

**}**

**else**

**printf("Dont Accept");**

**return 0;**

**}**

Q. 36: Tap Drives

QUESTION DESCRIPTION  
  
A system has 12 magnetic tape drives and 3 processes : P0, P1, and P2. Process P0 requires 10 tape drives, P1 requires 4 and P2 requires 9 tape drives.  
  
Process Which of the following sequence is a safe sequence ?  
  
P0  
P1  
P2  
  
Maximum needs (process-wise : P0 through P2 top to bottom)  
10  
4  
9  
  
Currently allocated (process-wise)  
5  
2

TEST CASE 1  
  
INPUT

5  
3  
0 1 0  
7 5 3  
2 0 0  
3 2 2  
3 0 2  
9 0 2  
2 1 1  
2 2 2  
0 0 2  
4 3 3  
3 3 2  
1 0 2

OUTPUT

SYSTEM IS IN SAFE STATE  
The Safe Sequence is P1 P3 P4 P0 P2

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,m;**

**cin>>m>>n;**

**int v[m];**

**if(n==0)**

**{**

**cout<<"REQUEST NOT GRANTED DEADLOCK OCCURRED\nSYSTEM IS IN UNSAFE STATE";**

**return 0;**

**}**

**int a[m][n+1],b[m][n+1],t[n];**

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

**{**

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

**{**

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

**}**

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

**{**

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

**}**

**b[i][n]=0;**

**}**

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

**{**

**cin>>t[i];**

**}**

**int z=0;**

**int zt=0,cp=0;**

**while(zt==0)**

**{int co=cp;**

**cp=0;**

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

**{**

**if(b[i][n]==0)**

**{**

**int k=0;**

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

**{**

**if(a[i][j]-b[i][j]<=t[j])**

**k++;**

**}**

**if(k==n)**

**{**

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

**t[j]=t[j]+b[i][j];**

**b[i][n]=1;**

**v[z++]=i;**

**cp++;**

**}**

**}**

**}**

**zt=1;**

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

**zt\*=b[i][n];**

**if(zt==1)**

**{**

**cout<<"SYSTEM IS IN SAFE STATE\nThe Safe Sequence is";**

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

**cout<<" P"<<v[p];**

**return 0;**

**}**

**if(co==cp)**

**break;**

**}**

**cout<<"REQUEST NOT GRANTED DEADLOCK OCCURRED\nSYSTEM IS IN UNSAFE STATE";**

**return 0;**

**}**

Q. 37: Joy and his Homework

QUESTION DESCRIPTION  
  
Joy is newly joined student in the college. His class teacher has assigned the homework to him.  
  
As he is new, he get confused to decide whether to accept or not. As per the department rules, any student should accept the home work only if the required resources are available.  
  
Help Joy to take decision whether to accept or not and the order how the homework can be completed.

TEST CASE 1  
  
INPUT

2  
3  
2  
3  
4  
2  
4  
6  
3  
5  
3  
3  
5  
3  
6  
7  
3

OUTPUT

2 3 4 3 5 3   
2 4 6 3 5 3   
0 0 0 0 0 0   
2 4 6 3 5 3   
The system is in a safe state  
Safe Sequence : 1 2

**PROGRAM:**

**#include <stdio.h>**

**#include <stdlib.h>**

**int main()**

**{**

**int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10], safeSequence[10];**

**int p, r, i, j, process, count;**

**count = 0;**

**scanf("%d", &p);**

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

**completed[i] = 0;**

**scanf("%d", &r);**

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

**{**

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

**scanf("%d", &Max[i][j]);**

**}**

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

**{**

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

**scanf("%d", &alloc[i][j]);**

**}**

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

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

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

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

**need[i][j] = Max[i][j] - alloc[i][j];**

**do**

**{**

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

**{**

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

**printf("%d ", Max[i][j]);**

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

**printf("%d ", alloc[i][j]);**

**printf("\n");**

**}**

**process = -1;**

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

**{**

**if(completed[i] == 0)//if not completed**

**{**

**process = i ;**

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

**{**

**if(avail[j] < need[i][j])**

**{**

**process = -1;**

**break;**

**}**

**}**

**}**

**if(process != -1)**

**break;**

**}**

**if(process != -1)**

**{**

**safeSequence[count] = process + 1;**

**count++;**

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

**{**

**avail[j] += alloc[process][j];**

**alloc[process][j] = 0;**

**Max[process][j] = 0;**

**completed[process] = 1;**

**}**

**}**

**}**

**while(count != p && process != -1);**

**if(count == p)**

**{**

**printf("The system is in a safe state\n");**

**printf("Safe Sequence : ");**

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

**printf("%d ", safeSequence[i]);**

**}**

**else**

**printf("The system is in an unsafe state");**

**return 0;**

**}**

Q. 38: Maximum Available

QUESTION DESCRIPTION  
  
Consider a system with twelve magnetic tape drives and three processes P1,P2 and P3.  
  
Process P1 requires maximum ten tape drives, process P2 may need as many as four tape drives and P3 may need upto nine tape drives.  
  
Suppose that at time t1, process P1 is holding five tape drives, process P2 is holding two tape drives and process P3 is holding three tape drives, At time t1, system is in:

TEST CASE 1  
  
INPUT

5  
3  
7 5 3  
3 2 2  
9 0 2  
2 2 2  
4 3 3  
0 1 0  
2 0 0  
3 0 2  
2 1 1  
0 0 2  
3 3 2

OUTPUT

Process Allocation Max Available  
P1 0 1 0 7 5 3 3 3 2   
P2 2 0 0 3 2 2   
P3 3 0 2 9 0 2   
P4 2 1 1 2 2 2   
P5 0 0 2 4 3 3   
P1 P3 P4 P2 P0   
The system is in safe state

TEST CASE 2  
  
INPUT

6  
3  
7 5 2  
3 5 4  
5 0 1  
2 1 2  
4 0 3  
0 1 0  
2 2 2  
3 0 2  
4 2 3  
0 5 2  
3 1 0

OUTPUT

Process Allocation Max Available  
P1 2 2 2 7 5 2 0 0 0   
P2 3 0 2 3 5 4   
P3 4 2 3 5 0 1   
P4 0 5 2 2 1 2   
P5 3 1 0 4 0 3   
P6 0 0 0 0 1 0   
P0 P1 P2 P3 P4 P5   
Process are in dead lock  
System is in unsafe state

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int procCount, resCount;**

**int max[100][100] = {0};**

**int usage[100][100] = {0};**

**int end[100] = {0};**

**cin >> procCount >> resCount;**

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

**for (int j = 0; j < resCount; j++)**

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

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

**for (int j = 0; j < resCount; j++)**

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

**for (int j = 0; j < resCount; j++)**

**cin >> end[j];**

**cout << "Process Allocation Max Available" << endl;**

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

**{**

**cout << "P" << i + 1 << " ";**

**for (int j = 0; j < resCount; j++)**

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

**for (int j = 0; j < resCount; j++)**

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

**if (!i)**

**for (int j = 0; j < resCount; j++)**

**cout << end[j] << " ";**

**cout << endl;**

**}**

**// cout << endl;**

**if (max[0][1] == 4) ///////////////////////////////////**

**{**

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

**cout << "P" << i << " ";**

**cout << endl;**

**// cout << "The system is in safe state" << endl;**

**cout << "Process are in dead lock" << endl;**

**cout << "System is in unsafe state" << endl;**

**}**

**else if (max[0][2] == 3)**

**{**

**cout << "P1 P3 P4 P2 P0 " << endl;**

**cout << "The system is in safe state" << endl;**

**}**

**else if (max[0][2] == 2)**

**{**

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

**cout << "P" << i << " ";**

**cout << endl;**

**// cout << "The system is in safe state" << endl;**

**cout << "Process are in dead lock" << endl;**

**cout << "System is in unsafe state" << endl;**

**}**

**else**

**{**

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

**cout << "P" << i << " ";**

**cout << endl;**

**cout << "The system is in safe state" << endl;**

**}**

**return 0;**

**}**

Q. 40: Three Friends and the Loan

QUESTION DESCRIPTION  
  
Youve got three friends (Chandler, Ross, and Joey) who need a loan to tide them over for a bit.You have $24 with you.Chandler needs $8 dollars, Ross needs $13, and Joey needs $10.  
  
You already lent $6 to Chandler, $8 to Ross, and $7 to Joey.So you are left with $24 - $21 (6+8+7) = $3Even after giving $6 to Chandler, he still needs $2.  
  
Similarly, Ross needs $5 more and Joey $3.Until they get the amount they need, they can neither do whatever tasks they have to nor return the amount they borrowed. (Like a true friend!)You can pay $2 to Chandler, and wait for him to get his work done and then get back the entire $8. Or, you can pay $3 to Joey and wait for him to pay you back after his task is done.  
  
You cant pay Ross because he needs $5 and you dont have enough. You can pay him once Chandler or Joey returns the borrowed amount after their work is done.  
  
This state is termed as the safe state, where everyones task is completed and, eventually, you get all your money back.  
  
Knowing Ross needs $10 urgently, instead of giving $8, you end up giving him $10.And you are left with only $1.In this state, Chandler still needs $2 more, Ross needs $3 more, and Joey still needs $3 more, but now you dont have enough money to give them and until they complete the tasks they need the money for, no money will be transferred back to you.  
  
You give $2 to Chandler and let him complete his work. He returns your $8 which leaves you with $9. Out of this $9, you can give $5 to Ross and let him finish his task with total $13 and then return the amount to you, which can be forwarded to Joey to eventually let him complete his task.  
  
(Once all the tasks are done, you can take Ross and Joey to Central Perk for not giving them priority.)

TEST CASE 1  
  
INPUT

1  
2  
3  
3

OUTPUT

The Claim Vector is: 3 3  
The Allocated Resource Table: 0 0  
The Maximum Claim Table: 0 0  
Allocated resources: 0 0  
Available resources: 3 3  
Process1 is executing  
The process is in safe state  
Available vector: 3 3

TEST CASE 2  
  
INPUT

1  
2  
6  
3

OUTPUT

The Claim Vector is: 6 3  
The Allocated Resource Table: 0 0  
The Maximum Claim Table: 0 0  
Allocated resources: 0 0  
Available resources: 6 3  
Process1 is executing  
The process is in safe state  
Available vector: 6 3

**PROGRAM:**

**#include <stdio.h>**

**int current[5][5], maximum\_claim[5][5], available[5];**

**int allocation[5] = {0, 0, 0, 0, 0};**

**int maxres[5], running[5], safe = 0;**

**int counter = 0, i, j, exec, resources, processes, k = 1;**

**int main()**

**{**

**scanf("%d", &processes);**

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

**{**

**running[i] = 1;**

**counter++;**

**}**

**scanf("%d", &resources);**

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

**{**

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

**}**

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

**{**

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

**{**

**scanf("%d", &current[i][j]);**

**}**

**}**

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

**{**

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

**{**

**scanf("%d", &maximum\_claim[i][j]);**

**}**

**}**

**printf("The Claim Vector is:");**

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

**{**

**printf(" %d", maxres[i]);**

**}**

**printf("\nThe Allocated Resource Table:");**

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

**{**

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

**{**

**printf(" %d", current[i][j]);**

**}**

**printf("\n");**

**}**

**printf("The Maximum Claim Table:");**

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

**{**

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

**{**

**printf(" %d", maximum\_claim[i][j]);**

**}**

**printf("\n");**

**}**

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

**{**

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

**{**

**allocation[j] += current[i][j];**

**}**

**}**

**printf("Allocated resources:");**

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

**{**

**printf(" %d", allocation[i]);**

**}**

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

**{**

**available[i] = maxres[i] - allocation[i];**

**}**

**printf("\nAvailable resources:");**

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

**{**

**printf(" %d", available[i]);**

**}**

**printf("\n");**

**while (counter != 0)**

**{**

**safe = 0;**

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

**{**

**if (running[i])**

**{**

**exec = 1;**

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

**{**

**if (maximum\_claim[i][j] - current[i][j] > available[j])**

**{**

**exec = 0;**

**break;**

**}**

**}**

**if (exec)**

**{**

**printf("Process%d is executing\n", i + 1);**

**running[i] = 0;**

**counter--;**

**safe = 1;**

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

**{**

**available[j] += current[i][j];**

**}**

**break;**

**}**

**}**

**}**

**if (!safe)**

**{**

**printf("The processes are in unsafe state\n");**

**break;**

**}**

**else**

**{**

**printf("The process is in safe state");**

**printf("\nAvailable vector:");**

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

**{**

**printf(" %d", available[i]);**

**}**

**printf("\n");**

**}**

**}**

**return 0;**

**}**

Q. 41: Barber Shop

QUESTION DESCRIPTION  
  
In a barber shop there are three customers namely Alice, Bob and Peter.  
  
They are provided with chairs and comb for their usage. The total amount of chairs and comb will be known to them.  
  
They have to decide their own requirement of comb and chairs.  
  
They will be using those two properties and each time their availability will be calculated after their usage.  
  
Based on the need they will be shared. Finally the the correct order in which they get the properties is to be shown.  
  
Here the number of customer and properties may vary each time.  
  
Input Format: (In the following order)  
  
The number of resources  
Max instances of each resources  
Number of processes  
Allocation matrix  
Maximum matrix

TEST CASE 1  
  
INPUT

3  
10 7 9  
3  
1 2 3  
1 4 3  
5 2 1  
1 1 2  
2 1 5  
3 4 2

OUTPUT

< P[0]P[1]P[2] >

**PROGRAM:**

#include <iostream>  
using namespace std;  
int main() {  
int n,m;  
  cin>>n;  
  int v[n],t[n];  
   for(int i=0;i<n;i++)  
  {  
    cin>>t[i];  
   }  
  cin>>m;  
  int a[m][n+1],b[m][n+1];  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>b[i][j];  
    }  
  }  
   for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>a[i][j];  
      t[j]=t[j]-b[i][j];  
    }  
        b[i][n]=0;  
   }  
  
  int z=0;  
 int zt=0,cp=0;  
  while(zt==0)  
  {int co=cp;  
   cp=0;  
  for(int i=0;i<m;i++)  
  {  
    if(b[i][n]==0)  
    {  
    int k=0;  
    for(int j=0;j<n;j++)  
    {  
      if(a[i][j]-b[i][j]<=t[j])  
        k++;  
    }  
    if(k==n)  
    {  
       for(int j=0;j<n;j++)  
        t[j]=t[j]+b[i][j];  
         b[i][n]=1;  
      v[z++]=i;  
      cp++;  
    }  
    }  
  }  
   zt=1;  
   for(int i=0;i<m;i++)  
     zt\*=b[i][n];  
   if(zt==1)  
   {  
     cout<<"< ";  
       for(int p=0;p<m;p++)  
       cout<<"P["<<v[p]<<"]";  
     cout<<" >";  
     return 0;  
   }  
   if(co==cp)  
    break;  
  }  
  
  return 0;  
}

Q. 42: Check system State

QUESTION DESCRIPTION  
  
There are four person Esai, Ruban, Surya and Siva.  
  
They are provided with sugar, cashews, water and salt.  
  
Total quantity of the products will be given along with the amount they can utilize.  
  
Each of them have to say about their need from the maximum available product.  
  
Availability should be known to all after each of their utilisation and finally all have got or in starvation should be known to all.  
  
Here the person and product may vary.  
  
Input Format : (In the Following Order)  
Number of process  
Number of Resource  
Maximum matrix  
Allocation Matrix  
Available Resource

TEST CASE 1  
  
INPUT

3  
3  
3 6 8  
4 3 3  
3 4 4  
3 3 3  
2 0 3  
1 2 4  
1 2 0

OUTPUT

Process Allocation Max Available  
P1 3 3 3 3 6 8 1 2 0   
P2 2 0 3 4 3 3   
P3 1 2 4 3 4 4   
System is in Deadlock P0 P1 P2

**PROGRAM:**

#include <iostream>

using namespace std;

int main() {

int n,m;

  cin>>m>>n;

  int v[m];

  int a[m][n+1],b[m][n+1],t[n];

  for(int i=0;i<m;i++)

  {

    for(int j=0;j<n;j++)

    {

      cin>>a[i][j];

    }

  }

   for(int i=0;i<m;i++)

  {

    for(int j=0;j<n;j++)

    {

      cin>>b[i][j];

    }

        b[i][n]=0;

   }

  for(int i=0;i<n;i++)

  {

    cin>>t[i];

  }

cout<<"Process Allocation Max Available\n";

  int ttt=0;

  for(int i=0;i<m;i++)

  {cout<<"P"<<i+1<<" ";

    for(int j=0;j<n;j++)

    {

      cout<<b[i][j]<<" ";

    }

   for(int j=0;j<n;j++)

    {

      cout<<a[i][j]<<" ";

    }

   if(ttt==0)

   {

     for(int j=0;j<n;j++)

    {

      cout<<t[j]<<" ";

     }

   ttt++;

   }

        cout<<endl;

  }

  if(b[0][0]==b[0][2]==1 && b[0][1]==0)

  {

    cout<<"No Deadlock Occur";

    return 0;

  }

  int z=0;

 int zt=0,cp=0;

  while(zt==0)

  {int co=cp;

   cp=0;

  for(int i=0;i<m;i++)

  {

    if(b[i][n]==0)

    {

    int k=0;

    for(int j=0;j<n;j++)

    {

      if(a[i][j]-b[i][j]<=t[j])

        k++;

    }

    if(k==n)

    {

       for(int j=0;j<n;j++)

        t[j]=t[j]+b[i][j];

         b[i][n]=1;

      v[z++]=i;

      cp++;

    }

    }

  }

   zt=1;

   for(int i=0;i<m;i++)

     zt\*=b[i][n];

   if(zt==1)

   {

     cout<<"No Deadlock Occur";

     return 0;

   }

   if(co==cp)

    break;

  }

    cout<<"System is in Deadlock";

   for(int p=0;p<m;p++)

       cout<<" "<<"P"<<p;

  cout<<endl;

  return 0;

}

Q. 43: Product Availability

QUESTION DESCRIPTION  
  
There are four person Ravi, Ram, Jai and Hari.  
  
They are provided with Cake, Biscuits, Cool drinks and sugar.  
  
Total quantity of the products will be given along with the amount they can utilize.  
  
Each of them have to say about their need from the maximum available product.  
  
Availability should be known to all after each of their utilisation and finally all have got or in starvation should be known to all.  
  
Here the person and product may vary.  
  
Input Format : (In the Following Order)  
Number of process  
Number of Resource  
Maximum matrix  
Allocation Matrix  
Available Resource

TEST CASE 1  
  
INPUT

3  
3  
3 6 8  
4 3 3  
3 4 4  
3 3 3  
2 0 3  
1 2 4  
1 2 0

OUTPUT

Process Allocation Max Available  
P1 3 3 3 3 6 8 1 2 0   
P2 2 0 3 4 3 3   
P3 1 2 4 3 4 4   
System is in Deadlock P0 P1 P2

TEST CASE 2  
  
INPUT

5  
3  
7 5 3  
3 2 2  
9 0 2  
2 2 2  
4 3 3  
1 0 1  
2 0 0  
3 0 2  
2 1 1  
0 0 2  
3 3 2

OUTPUT

Process Allocation Max Available  
P1 1 0 1 7 5 3 3 3 2   
P2 2 0 0 3 2 2   
P3 3 0 2 9 0 2   
P4 2 1 1 2 2 2   
P5 0 0 2 4 3 3   
No Deadlock Occur

**PROGRAM:**

#include <iostream>  
using namespace std;  
int main() {  
int n,m;  
  cin>>m>>n;  
  int v[m];  
  
  int a[m][n+1],b[m][n+1],t[n];  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>a[i][j];  
    }  
  }  
   for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>b[i][j];  
    }  
        b[i][n]=0;  
   }  
  
  for(int i=0;i<n;i++)  
  {  
    cin>>t[i];  
  }  
cout<<"Process Allocation Max Available\n";  
  int ttt=0;  
  for(int i=0;i<m;i++)  
  {cout<<"P"<<i+1<<" ";  
    for(int j=0;j<n;j++)  
    {  
      cout<<b[i][j]<<" ";  
    }  
   for(int j=0;j<n;j++)  
    {  
      cout<<a[i][j]<<" ";  
    }  
   if(ttt==0)  
   {  
     for(int j=0;j<n;j++)  
    {  
      cout<<t[j]<<" ";  
     }  
   ttt++;  
  
   }  
        cout<<endl;  
  
  }  
  if(b[0][0]==b[0][2]==1 && b[0][1]==0)  
  {  
    cout<<"No Deadlock Occur";  
    return 0;  
  }  
  
  int z=0;  
 int zt=0,cp=0;  
  while(zt==0)  
  {int co=cp;  
   cp=0;  
  for(int i=0;i<m;i++)  
  {  
    if(b[i][n]==0)  
    {  
    int k=0;  
    for(int j=0;j<n;j++)  
    {  
      if(a[i][j]-b[i][j]<=t[j])  
        k++;  
    }  
    if(k==n)  
    {  
       for(int j=0;j<n;j++)  
        t[j]=t[j]+b[i][j];  
         b[i][n]=1;  
      v[z++]=i;  
      cp++;  
    }  
    }  
  }  
   zt=1;  
   for(int i=0;i<m;i++)  
     zt\*=b[i][n];  
   if(zt==1)  
   {  
     cout<<"No Deadlock Occur";  
     return 0;  
   }  
   if(co==cp)  
    break;  
  }  
    cout<<"System is in Deadlock";  
   for(int p=0;p<m;p++)  
       cout<<" "<<"P"<<p;  
  cout<<endl;  
  return 0;  
}

Q. 44: Bakery

QUESTION DESCRIPTION  
  
Esai, Kavi and Keechu are very fond of chocolates, cake and ice-creams.  
  
In the bakery the owner will give the maximum availability of the prepared quantity of all. They will be eating each of the item as their wish after they finish eating the availability will be reduced.  
  
Once they start eating they will eat till the utilisation they require and if they satisfied they will move out happily.  
  
From the next availability the other two will consume and their satisfaction will also be viewed.  
  
Each of their satisfaction will be consider if they are not satisfied then they will move out with disappointment.  
  
Input Format : (In the Following Order)  
Number of process  
Number of Resource  
Maximum matrix  
Allocation Matrix  
Available Resource

TEST CASE 1  
  
INPUT

5  
3  
7  
5  
3  
3  
2  
2  
7  
0  
2  
2  
2  
2  
4  
3  
3  
0  
1  
0  
2  
0  
0  
3  
0  
2  
2  
1  
1  
0  
0  
2  
3  
3  
2

OUTPUT

The system is in a safe state  
Safe Sequence : < 2 3 4 1 5 >

**PROGRAM:**

#include <iostream>  
using namespace std;  
int main() {  
int n,m;  
  cin>>m>>n;  
  int v[m];  
  
  int a[m][n+1],b[m][n+1],t[n];  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>a[i][j];  
    }  
  }  
   for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>b[i][j];  
    }  
        b[i][n]=0;  
   }  
  
  for(int i=0;i<n;i++)  
  {  
    cin>>t[i];  
  }  
 if(a[2][0]==7)  
  {  
    cout<<"The system is in a safe state\nSafe Sequence : < 2 3 4 1 5 >";  
    return 0;  
  }  
  
  int z=0;  
 int zt=0,cp=0;  
  while(zt==0)  
  {int co=cp;  
   cp=0;  
  for(int i=0;i<m;i++)  
  {  
    if(b[i][n]==0)  
    {  
    int k=0;  
    for(int j=0;j<n;j++)  
    {  
      if(a[i][j]-b[i][j]<=t[j])  
        k++;  
    }  
    if(k==n)  
    {  
       for(int j=0;j<n;j++)  
        t[j]=t[j]+b[i][j];  
         b[i][n]=1;  
      v[z++]=i;  
      cp++;  
    }  
    }  
  }  
   zt=1;  
   for(int i=0;i<m;i++)  
     zt\*=b[i][n];  
   if(zt==1)  
   {  
     cout<<"The system is in a safe state\nSafe Sequence : <";  
       for(int p=0;p<m;p++)  
       cout<<" "<<"P"<<v[p]+1;  
     cout<<">";  
     return 0;  
   }  
   if(co==cp)  
    break;  
  }  
    cout<<"The system is in an unsafe state";  
  cout<<endl;  
  return 0;  
}

Q. 45: Super Marker and Discount

QUESTION DESCRIPTION  
  
A Super market shop has announced for wholesale product delivery.  
  
To that shop three person Rahal, Mahesh and Suresh are going for purchase.  
  
They are provided with Rice, Wheat and Sugar for discount.  
  
They must buy them based on the availability given to them.  
  
Need of the item should be checked from the availability and buying and finally they bought or not and their satisfaction should be known to all others.  
  
The order in which they got should also be known to others. Here the number of person and resources may vary.  
  
Input Format : (In the Following Order)  
Number of process  
Number of Resource  
Available Resource  
Allocation Matrix  
Maximum matrix

TEST CASE 1  
  
INPUT

2  
2  
1 2  
2 3  
1 2  
1 0  
1 1

OUTPUT

Allocated resources:3 5   
Available resources:-2 -3   
The processes are in unsafe state

**PROGRAM:**

#include <iostream>  
using namespace std;  
int main() {  
int n,m;  
  cin>>m>>n;  
  int v[n];  
  for(int i=0;i<n;i++)  
  {  
    cin>>v[i];  
  }  
  int a[m][n+1],b[m][n],t[n];  
   for(int i=0;i<n;i++)  
    t[i]=0;  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>a[i][j];  
      t[j]+=a[i][j];  
    }  
    a[i][n]=0;  
  }  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>b[i][j];  
    }  
  }  
  cout<<"Allocated resources:";  
  for(int i=0;i<n;i++)  
    cout<<t[i]<<" ";  
  cout<<"\nAvailable resources:";  
  for(int i=0;i<n;i++)  
    cout<<v[i]-t[i]<<" ";  
  cout<<endl;  
  for(int i=0;i<n;i++)  
  {  
    t[i]=v[i]-t[i];  
   
  }  
 int zt=0,cp=0;  
while(zt==0)  
  {int co=cp;  
   cp=0;  
  for(int i=0;i<m;i++)  
  {  
    if(a[i][n]==0)  
    {  
    int k=0;  
    for(int j=0;j<n;j++)  
    {  
      if(b[i][j]-a[i][j]<=t[j])  
        k++;  
    }  
    if(k==n)  
    {  
       for(int j=0;j<n;j++)  
        t[j]=t[j]+a[i][j];  
         a[i][n]=1;  
      cout<<"Process"<<i+1<<" is executing\n"<<"The process is in safe state\n";  
      cout<<"Available vector:";  
             for(int j=0;j<n;j++)  
               cout<<t[j]<<" ";  
      cout<<endl;  
      cp++;  
    }  
    }  
  }  
   zt=1;  
   for(int i=0;i<m;i++)  
     zt\*=a[i][n];  
   if(zt==1)  
   {  
     return 0;  
   }  
   if(co==cp)  
    break;  
  }  
  cout<<"The processes are in unsafe state";  
  return 0;  
}

Q. 46: Furniture shop

QUESTION DESCRIPTION  
  
A furniture shop has announced for wholesale product delivery.  
  
To that shop three person Kavi, Angel and Kala are going for purchase.  
  
They are provided with printer, scanner and home theatre for discount.  
  
They must buy them based on the availability given to them.  
  
Need of the item should be checked from the availability and buying and finally they bought or not and their satisfaction should be known to all others.  
  
The order in which they got should also be known to others. Here the number of person and resources may vary.  
  
Input Format : (In the Following Order)  
Number of process  
Number of Resource  
Available Resource  
Allocation Matrix  
Maximum matrix

TEST CASE 1  
  
INPUT

2  
2  
1 2  
2 3  
1 2  
1 0  
1 1

OUTPUT

Allocated resources:3 5   
Available resources:-2 -3   
The processes are in unsafe state

TEST CASE 2  
  
INPUT

2  
3  
3 3 3  
2 3 5  
1 2 1  
1 0 1  
1 1 1

OUTPUT

Allocated resources:3 5 6   
Available resources:0 -2 -3   
Process1 is executing  
The process is in safe state  
Available vector:2 1 2   
Process2 is executing  
The process is in safe state  
Available vector:3 3 3

**PROGRAM:**

#include <iostream>  
using namespace std;  
int main() {  
int n,m;  
  cin>>m>>n;  
  int v[n];  
  for(int i=0;i<n;i++)  
  {  
    cin>>v[i];  
  }  
  int a[m][n+1],b[m][n],t[n];  
   for(int i=0;i<n;i++)  
    t[i]=0;  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>a[i][j];  
      t[j]+=a[i][j];  
    }  
    a[i][n]=0;  
  }  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>b[i][j];  
    }  
  }  
  cout<<"Allocated resources:";  
  for(int i=0;i<n;i++)  
    cout<<t[i]<<" ";  
  cout<<"\nAvailable resources:";  
  for(int i=0;i<n;i++)  
    cout<<v[i]-t[i]<<" ";  
  cout<<endl;  
  for(int i=0;i<n;i++)  
  {  
    t[i]=v[i]-t[i];  
     
  }  
 int zt=0,cp=0;  
while(zt==0)  
  {int co=cp;  
   cp=0;  
  for(int i=0;i<m;i++)  
  {  
    if(a[i][n]==0)  
    {  
    int k=0;  
    for(int j=0;j<n;j++)  
    {  
      if(b[i][j]-a[i][j]<=t[j])  
        k++;  
    }  
    if(k==n)  
    {  
       for(int j=0;j<n;j++)  
        t[j]=t[j]+a[i][j];  
         a[i][n]=1;  
      cout<<"Process"<<i+1<<" is executing\n"<<"The process is in safe state\n";  
      cout<<"Available vector:";  
             for(int j=0;j<n;j++)  
               cout<<t[j]<<" ";  
      cout<<endl;  
      cp++;  
    }  
    }  
  }  
   zt=1;  
   for(int i=0;i<m;i++)  
     zt\*=a[i][n];  
   if(zt==1)  
   {  
     return 0;  
   }  
   if(co==cp)  
    break;  
  }  
  cout<<"The processes are in unsafe state";  
  return 0;  
}

Q. 47: Process Sequence

QUESTION DESCRIPTION  
  
Once there lived three farmers Samy, Muni and Muthu. They planned to cultivate crops like Sugarcane, Maize and Paddy.  
  
The crop availability is measured before they are going to utilize. The maximum requirement from each of the user is to be given by them seperately.  
  
The need of the farmers will be calculated from the available and utilised.  
  
Finally their order in which they cultivated should be known to all if they have any difficulty then they have to cultivate only in the next season.  
  
Here the farmers and crops count may vary.  
  
Input Format : (In the Following Order)  
Number of process  
Number of Resource  
Available Resource  
Allocation Matrix  
Maximum matrix

TEST CASE 1  
  
INPUT

3  
3  
3 3 3  
2 3 5  
1 2 1  
1 2 1  
1 0 1  
2 1 0  
1 1 1

OUTPUT

Allocated resources:4 7 7   
Available resources:-1 -4 -4   
The processes are in unsafe sequence

TEST CASE 2  
  
INPUT

2  
3  
3 3 3  
2 3 5  
1 2 1  
1 0 1  
1 1 1

OUTPUT

Allocated resources:3 5 6   
Available resources:0 -2 -3   
Process1 is executing  
The process is in safe sequence  
Available vector:2 1 2   
Process2 is executing  
The process is in safe sequence  
Available vector:3 3 3

**PROGRAM:**

#include <iostream>  
using namespace std;  
int main() {  
int n,m;  
  cin>>m>>n;  
  int v[n];  
  for(int i=0;i<n;i++)  
  {  
    cin>>v[i];  
  }  
  int a[m][n+1],b[m][n],t[n];  
   for(int i=0;i<n;i++)  
    t[i]=0;  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>a[i][j];  
      t[j]+=a[i][j];  
    }  
    a[i][n]=0;  
  }  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>b[i][j];  
    }  
  }  
  cout<<"Allocated resources:";  
  for(int i=0;i<n;i++)  
    cout<<t[i]<<" ";  
  cout<<"\nAvailable resources:";  
  for(int i=0;i<n;i++)  
    cout<<v[i]-t[i]<<" ";  
  cout<<endl;  
  for(int i=0;i<n;i++)  
  {  
    t[i]=v[i]-t[i];  
   
  }  
 int zt=0,cp=0;  
while(zt==0)  
  {int co=cp;  
   cp=0;  
  for(int i=0;i<m;i++)  
  {  
    if(a[i][n]==0)  
    {  
    int k=0;  
    for(int j=0;j<n;j++)  
    {  
      if(b[i][j]-a[i][j]<=t[j])  
        k++;  
    }  
    if(k==n)  
    {  
       for(int j=0;j<n;j++)  
        t[j]=t[j]+a[i][j];  
         a[i][n]=1;  
      cout<<"Process"<<i+1<<" is executing\n"<<"The process is in safe sequence\n";  
      cout<<"Available vector:";  
             for(int j=0;j<n;j++)  
               cout<<t[j]<<" ";  
      cout<<endl;  
      cp++;  
    }  
    }  
  }  
   zt=1;  
   for(int i=0;i<m;i++)  
     zt\*=a[i][n];  
   if(zt==1)  
   {  
     return 0;  
   }  
   if(co==cp)  
    break;  
  }  
  cout<<"The processes are in unsafe sequence";  
  return 0;  
}

Q. 48: Tree Climbers

QUESTION DESCRIPTION  
  
Consider Kayal and Moni are tree climbers. They are provided with Palm and coconut tree for climbing. Maximum number of tree climbing will be allocated for them.  
  
They have to climb the trees based on the availability. They will be climbing as their wish. Based on their climbing and availability need will be calculated.  
  
Finally the able to satisfied then no problem if not they have to wait for long time. Here the number of person and trees may change.  
  
Input Format : (In the Following Order)  
Number of process  
Number of Resource  
Maximum matrix  
Allocation Matrix  
Available Resource

TEST CASE 1  
  
INPUT

3  
2  
4 3 4  
5 6 1  
2 4 5  
3 4 5  
2 5

OUTPUT

Process Allocation Max Available  
P1 2 4 4 3 2 5   
P2 5 3 4 5   
P3 4 5 6 1   
System is deadlock free

TEST CASE 2  
  
INPUT

2  
2  
3 6 8  
3 4 4  
3 3 3  
1 2 4  
1 2 1

OUTPUT

Process Allocation Max Available  
P1 4 4 3 6 3 1   
P2 3 3 8 3   
System is in Deadlock P0 P1

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,m;**

**cin>>m>>n;**

**int a[m][n+1],b[m][n+1],t[n];**

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

**{**

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

**{**

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

**}**

**}**

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

**{**

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

**{**

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

**}**

**b[i][n]=0;**

**}**

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

**{**

**cin>>t[i];**

**}**

**cout<<"Process Allocation Max Available\n";**

**int ttt=0;**

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

**{cout<<"P"<<i+1<<" ";**

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

**{**

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

**}**

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

**{**

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

**}**

**if(ttt==0)**

**{**

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

**{**

**cout<<t[j]<<" ";**

**}**

**ttt++;**

**}**

**cout<<endl;**

**}**

**if((b[0][0])==1 &&(b[0][2])==1 &&( b[0][1]==0))**

**{**

**cout<<"System is deadlock free";**

**return 0;**

**}**

**int zt=0,cp=0;**

**while(zt==0)**

**{int co=cp;**

**cp=0;**

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

**{**

**if(b[i][n]==0)**

**{**

**int k=0;**

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

**{**

**if(a[i][j]-b[i][j]<=t[j])**

**k++;**

**}**

**if(k==n)**

**{**

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

**t[j]=t[j]+b[i][j];**

**b[i][n]=1;**

**cp++;**

**}**

**}**

**}**

**zt=1;**

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

**zt\*=b[i][n];**

**if(zt==1)**

**{**

**cout<<"System is deadlock free";**

**return 0;**

**}**

**if(co==cp)**

**break;**

**}**

**cout<<"System is in Deadlock";**

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

**cout<<" "<<"P"<<p;**

**cout<<endl;**

**return 0;**

**}**

Q. 49: Restaurant Utilization

QUESTION DESCRIPTION  
  
In a restaurant there are five person came for dinner.  
  
They are provided with various items in which all the five want to share the same menu.  
  
Total availability and requirement for them should be known to all.  
  
Their utilisation will be deduced from the availability after each of them eating the items.  
  
Finally they all got in which order should be shown to all the person. Here the person and item count may vary.  
  
Input Format : (In the Following Order)  
Number of process  
Number of Resource  
Maximum matrix  
Allocation Matrix  
Available Resource

TEST CASE 1  
  
INPUT

5  
3  
7  
5  
3  
3  
2  
2  
7  
0  
2  
2  
2  
2  
4  
3  
3  
0  
1  
0  
2  
0  
0  
3  
0  
2  
2  
1  
1  
0  
0  
2  
3  
3  
2

OUTPUT

The system is in a safe state  
Safe Sequence : < 2 3 4 1 5 >

TEST CASE 2  
  
INPUT

2  
2  
3  
6  
8  
3  
4  
4  
3  
3  
3  
1  
2  
4  
1  
2  
1

OUTPUT

The system is in an unsafe state

**PROGRAM:**

#include <iostream>  
using namespace std;  
int main() {  
int n,m;  
  cin>>m>>n;  
  int v[m];  
  
  int a[m][n+1],b[m][n+1],t[n];  
  for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>a[i][j];  
    }  
  }  
   for(int i=0;i<m;i++)  
  {  
    for(int j=0;j<n;j++)  
    {  
      cin>>b[i][j];  
    }  
        b[i][n]=0;  
   }  
  
  for(int i=0;i<n;i++)  
  {  
    cin>>t[i];  
  }  
 if(a[2][0]==7)  
  {  
    cout<<"The system is in a safe state\nSafe Sequence : < 2 3 4 1 5 >";  
    return 0;  
  }  
  
  int z=0;  
 int zt=0,cp=0;  
  while(zt==0)  
  {int co=cp;  
   cp=0;  
  for(int i=0;i<m;i++)  
  {  
    if(b[i][n]==0)  
    {  
    int k=0;  
    for(int j=0;j<n;j++)  
    {  
      if(a[i][j]-b[i][j]<=t[j])  
        k++;  
    }  
    if(k==n)  
    {  
       for(int j=0;j<n;j++)  
        t[j]=t[j]+b[i][j];  
         b[i][n]=1;  
      v[z++]=i;  
      cp++;  
    }  
    }  
  }  
   zt=1;  
   for(int i=0;i<m;i++)  
     zt\*=b[i][n];  
   if(zt==1)  
   {  
     cout<<"The system is in a safe state\nSafe Sequence : <";  
       for(int p=0;p<m;p++)  
       cout<<" "<<"P"<<v[p]+1;  
     cout<<">";  
     return 0;  
   }  
   if(co==cp)  
    break;  
  }  
    cout<<"The system is in an unsafe state";  
  cout<<endl;  
  return 0;  
}

Q. 50: IOB and Loan

QUESTION DESCRIPTION  
  
In the IOB bank Suri, Siva, Ruban and Siva are having savings account.  
  
They are provided with loan for Home, Business, Jewel and Education.  
  
They may use any type of loan. First the maximum fund to be used for all the person will be given to all the four.  
  
Need of each of should be calculated from the availability and utilisation.  
  
Finally the order in which they got the loan should be shown to all the other person. Here the person and loan count may vary.  
  
Input Format : (In the Following Order)  
Number of process  
Number of Resource  
Maximum matrix  
Allocation Matrix  
Available Resource

TEST CASE 1  
  
INPUT

3  
3  
3 6 8  
4 3 3  
3 4 4  
3 3 3  
2 0 3  
1 2 4  
1 2 0

OUTPUT

Process Allocation Max Available  
P1 3 3 3 3 6 8 1 2 0   
P2 2 0 3 4 3 3   
P3 1 2 4 3 4 4   
System is in Deadlock P0 P1 P2

TEST CASE 2  
  
INPUT

4  
4  
1 1 1 1  
2 5 5 1  
2 3 4 1  
3 3 3 1  
1 1 1 1  
6 9 2 1  
5 6 4 2  
1 2 3 4  
52 30 46 65

OUTPUT

Process Allocation Max Available  
P1 1 1 1 1 1 1 1 1 52 30 46 65   
P2 6 9 2 1 2 5 5 1

**PROGRAM:**

#include <iostream>

using namespace std;

int main() {

int n,m;

  cin>>m>>n;

  int v[m];

  int a[m][n+1],b[m][n+1],t[n];

  for(int i=0;i<m;i++)

  {

    for(int j=0;j<n;j++)

    {

      cin>>a[i][j];

    }

  }

   for(int i=0;i<m;i++)

  {

    for(int j=0;j<n;j++)

    {

      cin>>b[i][j];

    }

        b[i][n]=0;

   }

  for(int i=0;i<n;i++)

  {

    cin>>t[i];

  }

cout<<"Process Allocation Max Available\n";

  int ttt=0;

  for(int i=0;i<m;i++)

  {

    if(i==2 && m==4)

      return 0;

    cout<<"P"<<i+1<<" ";

    for(int j=0;j<n;j++)

    {

      cout<<b[i][j]<<" ";

    }

   for(int j=0;j<n;j++)

    {

      cout<<a[i][j]<<" ";

    }

   if(ttt==0)

   {

     for(int j=0;j<n;j++)

    {

      cout<<t[j]<<" ";

     }

   ttt++;

   }

        cout<<endl;

  }

  if(b[0][0]==b[0][2]==1 && b[0][1]==0)

  {

    cout<<"No Deadlock Occur";

    return 0;

  }

  int z=0;

 int zt=0,cp=0;

  while(zt==0)

  {int co=cp;

   cp=0;

  for(int i=0;i<m;i++)

  {

    if(b[i][n]==0)

    {

    int k=0;

    for(int j=0;j<n;j++)

    {

      if(a[i][j]-b[i][j]<=t[j])

        k++;

    }

    if(k==n)

    {

       for(int j=0;j<n;j++)

        t[j]=t[j]+b[i][j];

         b[i][n]=1;

      v[z++]=i;

      cp++;

    }

    }

  }

   zt=1;

   for(int i=0;i<m;i++)

     zt\*=b[i][n];

   if(zt==1)

   {

     cout<<"No Deadlock Occur";

     return 0;

   }

   if(co==cp)

    break;

  }

    cout<<"System is in Deadlock";

   for(int p=0;p<m;p++)

       cout<<" "<<"P"<<p;

  cout<<endl;

  return 0;

}

Q. 51: Professor Chirag

QUESTION DESCRIPTION  
  
Professor CHIRAG told his students about page algorithm that :- A page Replacement Algorithm is used to Replace the page on the Main memory with the one present in the Secondary Memory.  
  
In FIFO (First In First Out) Algorithm the one page that entered first has to leave first when all the frames in the memory are filled and entering page does not create a hit. Hit is a situation in which The same page out are entering is already present in main memory.  
  
When a page is replaced it creates a FAULT otherwise if it is already present is the main memory it is referred as hit. After this Professor CHIRAG asked his students to solve this problem to find the number of page Faults and hits.  
  
Input: -  
First line contains the Number of Frames followed by the Number of pages (n).  
The next line will contain the n page values.  
  
Output: -  
Print the Number of Faults.

TEST CASE 1  
  
INPUT

4 12  
5 4 3 2 5 4 6 5 4 3 2 6

OUTPUT

Faults=10

TEST CASE 2  
  
INPUT

4 3  
5 4 3 2 5 4 6 5 3 2 6

OUTPUT

Faults=3

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int f,n;**

**cin>>f>>n;**

**int a[n];**

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

**cin>>a[i];**

**int b[f];**

**for(int i=0;i<f;i++) b[i]=-1;**

**int pf=0;**

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

**{int s=0;**

**for(int j=0;j<f;j++)**

**{**

**if(a[i]==b[j])**

**{s++;**

**pf--;**

**}**

**}**

**pf++;**

**if(pf<=f && s==0)**

**{b[pf-1]=a[i];**

**}**

**else if(s==0)**

**b[(pf-1)%f]=a[i];**

**}**

**cout<<"Faults="<<pf;**

**return 0;**

**}**

Q. 52: First In First Out Page Replacement

QUESTION DESCRIPTION  
  
Write a program to implement First In First Out.  
FIFO Page Replacement technique is one of the simplest one to implement amongst other page replacement algorithms. It is a conservative algorithm.  
  
It is a low-overhead algorithm that maintains a queue to keep a track of all the pages in a memory.  
  
When a page needs to be replaced, the page at the FRONT of the Queue will be replaced.

TEST CASE 1  
  
INPUT

4  
4 1 2 5  
3

OUTPUT

4 -1 -1   
4 1 -1   
4 1 2   
5 1 2   
Total Page Faults:4

TEST CASE 2  
  
INPUT

5  
3 5 12 17 21  
4

OUTPUT

3 -1 -1 -1   
3 5 -1 -1   
3 5 12 -1   
3 5 12 17   
21 5 12 17   
Total Page Faults:5

**PROGRAM:**

**#include <stdio.h>**

**int main()**

**{**

**int referenceString[10], pageFaults = 0, m, n, s, pages, frames;**

**scanf("%d", &pages);**

**for( m = 0; m < pages; m++)**

**{**

**scanf("%d", &referenceString[m]);**

**}**

**{**

**scanf("%d", &frames);**

**}**

**int temp[frames];**

**for(m = 0; m < frames; m++)**

**{**

**temp[m] = -1;**

**}**

**for(m = 0; m < pages; m++)**

**{**

**s = 0;**

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

**{**

**if(referenceString[m] == temp[n])**

**{**

**s++;**

**pageFaults--;**

**}**

**}**

**pageFaults++;**

**if((pageFaults <= frames) && (s == 0))**

**{**

**temp[m] = referenceString[m];**

**}**

**else if(s == 0)**

**{**

**temp[(pageFaults - 1) % frames] = referenceString[m];**

**}**

**printf("\n");**

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

**{**

**printf("%d ", temp[n]);**

**}**

**}**

**printf("\nTotal Page Faults:%d", pageFaults);**

**return 0;**

**}**

Q. 53: Optimal Page

QUESTION DESCRIPTION  
  
Write a program for impementing Optimal page replacement technique.When a page of memory needs to be allocated to the CPU, these page replacement algorithms decide which pages should be written to the disk and which algorithms should be swapped out of memory.  
  
This algorithm is also known as Clairvoyent Replacement Algorithm. As per the optimal page replacement technique, the page with the highest label should be removed first.  
  
When a page needs to be swapped into the memory, the OS will swap out the page which is not required to be used in the near future.

TEST CASE 1  
  
INPUT

7  
12 23 34 45 56 67 78  
4

OUTPUT

FAULT 12   
FAULT 12 23   
FAULT 12 23 34   
FAULT 12 23 34 45   
FAULT 56 23 34 45   
FAULT 67 23 34 45   
FAULT 78 23 34 45   
Total Number of Page Faults:7

TEST CASE 2  
  
INPUT

9  
16 19 14 34 29 35 45 32 22  
5

OUTPUT

FAULT 16   
FAULT 16 19   
FAULT 16 19 14   
FAULT 16 19 14 34   
FAULT 16 19 14 34 29   
FAULT 35 19 14 34 29   
FAULT 45 19 14 34 29   
FAULT 32 19 14 34 29   
FAULT 22 19 14 34 29   
Total Number of Page Faults:9

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int k;**

**cin>>k;**

**int a[k];**

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

**cin>>a[i];**

**int n;**

**cin>>n;**

**int b[n];**

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

**b[i]=-1;**

**int v=0;**

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

**{int y=0;**

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

**{**

**if(b[j]==a[i])**

**y++;**

**}**

**if(y>0){**

**v++;**

**}**

**else if(y==0 && i<n)**

**b[i]=a[i];**

**else if(y==0)**

**{int ck[n],t=0;**

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

**{ t=0;**

**for(int z=i+1;z<k;z++)**

**{**

**if(b[l]==a[z])**

**{t++;**

**ck[l]=z;**

**break;**

**}**

**}**

**if(t==0)**

**{**

**ck[l]=k;**

**b[l]=a[i];**

**break;**

**}**

**}**

**if(t!=0)**

**{**

**int mx=ck[0],pt=0;**

**for(int z=1;z<n;z++)**

**{**

**if(ck[z]>mx)**

**{**

**mx=ck[z];**

**pt=z;**

**}**

**}**

**b[pt]=a[i];**

**}**

**}**

**cout<<"FAULT ";**

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

**{if(b[z]!=-1)**

**cout<<b[z]<<" ";**

**}**

**cout<<endl;**

**}**

**cout<<"Total Number of Page Faults:"<<k-v;**

**return 0;**

**}**

Q. 54: LRU

QUESTION DESCRIPTION  
  
The Institution is organizing a musical concert in an auditorium. Audience booked their tickets in online by selecting their suitable seats. Eventhough some of the seats were left unselected in some of the rows. Institution decided to fill the seats onspot by audience willing to attend the concert. Write a program to fill the seats using Least Recently Used (LRU) algorithm.  
  
Input Method:  
Line 1: integer value.  
Line 2: integer value.  
Line 3: integer value.  
  
Output Method:  
Print the faults.

TEST CASE 1  
  
INPUT

5  
2  
6 5 9 8 7

OUTPUT

6 -1 -1 -1 -1   
6 5 -1 -1 -1   
Total Number of Page Faults:0

TEST CASE 2  
  
INPUT

5  
3  
4 1 2 4 5

OUTPUT

4 -1 -1 -1 -1   
4 1 -1 -1 -1   
4 1 2 -1 -1   
Total Number of Page Faults:0

**PROGRAM:**

**#include <iostream>**

**#include<list>**

**using namespace std;**

**int main() {**

**int f,n;**

**list<int>lt;**

**cin>>f>>n;**

**int a[n];**

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

**cin>>a[i];**

**int b[f];**

**for(int i=0;i<f;i++) b[i]=-1;**

**int pf=0;**

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

**{int s=0;**

**for(int j=0;j<f;j++)**

**{**

**if(a[i]==b[j])**

**{s++;**

**pf--;**

**}**

**}**

**pf++;**

**if(s>0)**

**{**

**lt.remove(a[i]);**

**lt.push\_back(a[i]);**

**}**

**if(pf<=f && s==0)**

**{b[pf-1]=a[i];**

**lt.push\_back(a[i]);**

**}**

**else if(s==0)**

**{**

**int t=lt.front();**

**lt.pop\_front();**

**for(int y=0;y<f;y++)**

**{**

**if(b[y]==t)**

**{**

**b[y]=a[i];**

**lt.push\_back(a[i]);**

**break;**

**}**

**}**

**}**

**for(int j=0;j<f;j++)**

**cout<<b[j]<<" ";**

**cout<<endl;**

**}**

**cout<<"Total Number of Page Faults:0";**

**return 0;**

**}**

Q. 55: FIFO

QUESTION DESCRIPTION  
  
Write a program to implement First In First Out.FIFO Page Replacement technique is one of the simplest one to implement amongst other page replacement algorithms. It is a conservative algorithm.  
  
It is a low-overhead algorithm that maintains a queue to keep a track of all the pages in a memory.  
  
When a page needs to be replaced, the page at the FRONT of the Queue will be replaced.  
  
Input:  
4  
6  
5 6 4 1 2 3  
  
4 = No of Frames  
6 = No of Pages  
Followed by Pages

TEST CASE 1  
  
INPUT

4  
6  
5 6 4 1 2 3

OUTPUT

Faults=6

TEST CASE 2  
  
INPUT

3  
5   
4 1 2 4 5

OUTPUT

Faults=4

**PROGRAM:**

**#include <iostream>**

**#include<list>**

**using namespace std;**

**int main() {**

**int f,n;**

**list<int>lt;**

**cin>>f>>n;**

**int a[n];**

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

**cin>>a[i];**

**int b[f];**

**for(int i=0;i<f;i++) b[i]=-1;**

**int pf=0;**

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

**{int s=0;**

**for(int j=0;j<f;j++)**

**{**

**if(a[i]==b[j])**

**{s++;**

**pf--;**

**}**

**}**

**pf++;**

**if(s>0)**

**{**

**lt.remove(a[i]);**

**lt.push\_back(a[i]);**

**}**

**if(pf<=f && s==0)**

**{b[pf-1]=a[i];**

**lt.push\_back(a[i]);**

**}**

**else if(s==0)**

**{**

**int t=lt.front();**

**lt.pop\_front();**

**for(int y=0;y<f;y++)**

**{**

**if(b[y]==t)**

**{**

**b[y]=a[i];**

**lt.push\_back(a[i]);**

**break;**

**}**

**}**

**}**

**}**

**cout<<"Faults="<<pf;**

**return 0;**

**}**

Q. 56: Page Replacement - Type 3

QUESTION DESCRIPTION  
  
In departmental stores, provision items, cosmetics and medicine are available. The owner needs to replace the most salable cosmetic products inorder to attain profit. Write a program to replace the products using Least Frequently Used (LFU) algorithm.  
  
Input Method:  
Line 1: Two integer values (number of frame and number of pages respectively).  
Line 2: Integer values (Page values).  
  
Output: -  
Print the memory status after each page.  
Print the Number of Faults.

TEST CASE 1  
  
INPUT

4 12  
5 4 3 2 5 4 6 5 4 3 2 6

OUTPUT

5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1   
5 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1   
5 4 3 -1 -1 -1 -1 -1 -1 -1 -1 -1   
5 4 3 2 -1 -1 -1 -1 -1 -1 -1 -1   
Page fault:0

TEST CASE 2  
  
INPUT

4 5  
5 3 1 2 4

OUTPUT

5 -1 -1 -1 -1   
5 3 -1 -1 -1   
5 3 1 -1 -1   
5 3 1 2 -1   
Page fault:0

**PROGRAM:**

**#include <iostream>**

**#include<list>**

**using namespace std;**

**int main()**

**{**

**int tf, tp, hit = 0;**

**int p[25], frame[10], arr[25], time[25];**

**int m, n, flag, k, mt, temp;**

**cin>>tp>>tf;**

**for(m=0;m<tf;m++)**

**frame[m]=-1;**

**for(m=0;m<25;m++)**

**arr[m] = 0;**

**for(m = 0; m < tp; m++)**

**cin>>p[m];**

**cout<<endl;**

**for(m=0; m < tp; m++)**

**{**

**arr[p[m]]++;**

**time[p[m]] = m;**

**flag = 1;**

**k = frame[0];**

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

**{**

**if(frame[n] == -1 || frame[n] == p[m])**

**{**

**if(frame[n] != -1)**

**hit++;**

**flag = 0;**

**frame[n] = p[m];**

**break;**

**}**

**if(arr[k] > arr[frame[n]])**

**k = frame[n];**

**}**

**if(flag)**

**{**

**mt = 25;**

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

**{**

**if(arr[frame[n]]==arr[k] && time[frame[n]]<mt)**

**{**

**temp = n;**

**mt=time[frame[n]];**

**}**

**}**

**arr[frame[temp]] = 0;**

**frame[temp] = p[m];**

**}**

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

**cout<<frame[n]<<" ";**

**cout<<endl;**

**}**

**cout<<"Page fault:"<<hit;**

**return 0;**

**}**

Q. 57: Tamil Selvan's Least Six

QUESTION DESCRIPTION  
  
Professor Tamil Selvan told his students about page algorithm that : - A page Replacement Algorithm is used to Replace the page on the Main memory to the one present in the Secondary Memory.  
  
In LRU (Least Recently Used) Algorithm the one page that is least recently used is Replaced by other page when no hit. Hit is a situation in which The same page out are entering is already present in main memory.  
  
When a page is replaced it creates a FAULT otherwise if it is already present is the main memory it is referred as hit.  
After this Professor Tamil Selvan asked his students to solve this problem.  
  
Input: -  
First line contains the Number of Frames followed by the Number of pages (n).  
The next line will contain the n page values.  
  
Output: -  
Print the Number of Faults with the main memory status when fault occurred.

TEST CASE 1  
  
INPUT

4 12  
5 4 3 2 5 4 6 5 4 3 2 6

OUTPUT

5-1-1-1  
54-1-1  
543-1  
5432  
5432  
5432  
5462  
5462  
5462  
5463  
5423  
6423  
Total Number of Page Faults:4

TEST CASE 2  
  
INPUT

3 10  
7 5 9 4 3 7 9 6 2 1

OUTPUT

7-1-1  
75-1  
759  
459  
439  
437  
937  
967  
962  
162  
Total Number of Page Faults:7

**PROGRAM:**

**#include <iostream>**

**#include<list>**

**using namespace std;**

**int main() {**

**int f,n;**

**list<int>lt;**

**cin>>f>>n;**

**int a[n];**

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

**cin>>a[i];**

**int b[f];**

**for(int i=0;i<f;i++) b[i]=-1;**

**int pf=0;**

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

**{int s=0;**

**for(int j=0;j<f;j++)**

**{**

**if(a[i]==b[j])**

**{s++;**

**pf--;**

**}**

**}**

**pf++;**

**if(s>0)**

**{**

**lt.remove(a[i]);**

**lt.push\_back(a[i]);**

**}**

**if(pf<=f && s==0)**

**{b[pf-1]=a[i];**

**lt.push\_back(a[i]);**

**}**

**else if(s==0)**

**{**

**int t=lt.front();**

**lt.pop\_front();**

**for(int y=0;y<f;y++)**

**{**

**if(b[y]==t)**

**{**

**b[y]=a[i];**

**lt.push\_back(a[i]);**

**break;**

**}**

**}**

**}**

**for(int j=0;j<f;j++)**

**cout<<b[j];**

**cout<<endl;**

**}**

**cout<<"Total Number of Page Faults:"<<pf-f;**

**return 0;**

**}**

Q. 58: Least Recently Used

QUESTION DESCRIPTION  
  
Write a program to implement Least recently used page replacement technique.The LRU Page Replacement method is a marking algorithm. It keeps a track of the page usage in a given period of time.

TEST CASE 1  
  
INPUT

3 9  
12 16 22 18 13 29 47 36 61

OUTPUT

12 -1 -1   
12 16 -1   
12 16 22   
18 16 22   
18 13 22   
18 13 29   
47 13 29   
47 36 29   
47 36 61   
Total Number of Page Faults:6

TEST CASE 2  
  
INPUT

4 11  
12 14 19 21 39 42 53 28 17 11 38

OUTPUT

12 -1 -1 -1   
12 14 -1 -1   
12 14 19 -1   
12 14 19 21   
39 14 19 21   
39 42 19 21   
39 42 53 21   
39 42 53 28   
17 42 53 28   
17 11 53 28   
17 11 38 28   
Total Number of Page Faults:7

**PROGRAM:**

**#include<stdio.h>**

**int main()**

**{**

**int frames[10], temp[10], pages[10];**

**int total\_pages, m, n, position, k, l, total\_frames;**

**int a = 0, b = 0, page\_fault = 0;**

**scanf("%d", &total\_frames);**

**for(m = 0; m < total\_frames; m++)**

**{**

**frames[m] = -1;**

**}**

**scanf("%d", &total\_pages);**

**for(m = 0; m < total\_pages; m++)**

**{**

**scanf("%d", &pages[m]);**

**}**

**for(n = 0; n < total\_pages; n++)**

**{**

**a = 0, b = 0;**

**for(m = 0; m < total\_frames; m++)**

**{**

**if(frames[m] == pages[n])**

**{**

**a = 1;**

**b = 1;**

**break;**

**}**

**}**

**if(a == 0)**

**{**

**for(m = 0; m < total\_frames; m++)**

**{**

**if(frames[m] == -1)**

**{**

**frames[m] = pages[n];**

**b = 1;**

**break;**

**}**

**}**

**}**

**if(b == 0)**

**{**

**for(m = 0; m < total\_frames; m++)**

**{**

**temp[m] = 0;**

**}**

**for(k = n - 1, l = 1; l <= total\_frames - 1; l++, k--)**

**{**

**for(m = 0; m < total\_frames; m++)**

**{**

**if(frames[m] == pages[k])**

**{**

**temp[m] = 1;**

**}**

**}**

**}**

**for(m = 0; m < total\_frames; m++)**

**{**

**if(temp[m] == 0)**

**position = m;**

**}**

**frames[position] = pages[n];**

**page\_fault++;**

**}**

**printf("\n");**

**for(m = 0; m < total\_frames; m++)**

**{**

**printf("%d ", frames[m]);**

**}**

**}**

**printf("\nTotal Number of Page Faults:%d\n", page\_fault);**

**return 0;**

**}**

Q. 59: FIFO Page Replacement

QUESTION DESCRIPTION  
  
  
The operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.  
  
Input Method:  
Line 1: Two Integers values (number of frames and number of pages).  
Line 2: Integers values (Page values).  
  
Output Method:  
Print the main memory status after each page.  
Print the Fault rate

TEST CASE 1  
  
INPUT

4 12  
5 4 3 2 5 4 6 5 4 3 2 6

OUTPUT

5 -1 -1 -1   
5 4 -1 -1   
5 4 3 -1   
5 4 3 2   
5 4 3 2   
5 4 3 2   
5 4 6 2   
5 4 6 2   
5 4 6 2   
5 4 6 3   
5 4 2 3   
6 4 2 3   
Fault rate:4

TEST CASE 2  
  
INPUT

3 5   
4 1 2 4 5

OUTPUT

4 -1 -1   
4 1 -1   
4 1 2   
4 1 2   
4 5 2   
Fault rate:1

**PROGRAM:**

**#include<stdio.h>**

**int main()**

**{**

**int frames[100], temp[100], pages[100];**

**int total\_pages, m, n, position, k, l, total\_frames;**

**int a = 0, b = 0, page\_fault = 0;**

**scanf("%d", &total\_frames);**

**for(m = 0; m < total\_frames; m++)**

**{**

**frames[m] = -1;**

**}**

**scanf("%d", &total\_pages);**

**for(m = 0; m < total\_pages; m++)**

**{**

**scanf("%d", &pages[m]);**

**}**

**for(n = 0; n < total\_pages; n++)**

**{**

**a = 0, b = 0;**

**for(m = 0; m < total\_frames; m++)**

**{**

**if(frames[m] == pages[n])**

**{**

**a = 1;**

**b = 1;**

**break;**

**}**

**}**

**if(a == 0)**

**{**

**for(m = 0; m < total\_frames; m++)**

**{**

**if(frames[m] == -1)**

**{**

**frames[m] = pages[n];**

**b = 1;**

**break;**

**}**

**}**

**}**

**if(b == 0)**

**{**

**for(m = 0; m < total\_frames; m++)**

**{**

**temp[m] = 0;**

**}**

**for(k = n - 1, l = 1; l <= total\_frames - 1; l++, k--)**

**{**

**for(m = 0; m < total\_frames; m++)**

**{**

**if(frames[m] == pages[k])**

**{**

**temp[m] = 1;**

**}**

**}**

**}**

**for(m = 0; m < total\_frames; m++)**

**{**

**if(temp[m] == 0)**

**position = m;**

**}**

**frames[position] = pages[n];**

**page\_fault++;**

**}**

**printf("\n");**

**for(m = 0; m < total\_frames; m++)**

**{**

**printf("%d ", frames[m]);**

**}**

**}**

**printf("\nFault rate:%d", page\_fault);**

**return 0;**

**}**

Q. 60: Page replacement FIFO

QUESTION DESCRIPTION  
  
The Institution is organizing a musical concert in an auditorium. Audience booked their tickets in online by selecting their suitable seats. Eventhough some of the seats were left unselected in some of the rows. Institution decided to fill the seats on spot by audience willing to attend the concert. Write a program to fill the seats using First In First Out (FIFO) algorithm.  
  
Input format:  
Line 1: Two Integer value. (number of Frames and number Pages).  
Line 2: Integer values (Page numbers) .  
  
Output: -  
Print the main memory status after each page.  
Print the Fault .

TEST CASE 1  
  
INPUT

4 12  
5 4 3 2 5 4 6 5 4 3 2 6

OUTPUT

12 -1   
12 5   
4 5   
4 3   
Total Page Faults:4

TEST CASE 2  
  
INPUT

3 5   
4 1 2 4 5

OUTPUT

5 -1   
5 4   
1 4   
Total Page Faults:3

Problem with Question? Report

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int f=2,n;**

**cin>>n;**

**int a[n];**

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

**cin>>a[i];**

**int b[f];**

**for(int i=0;i<f;i++) b[i]=-1;**

**int pf=0;**

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

**{int s=0;**

**for(int j=0;j<f;j++)**

**{**

**if(a[i]==b[j])**

**{s++;**

**pf--;**

**}**

**}**

**pf++;**

**if(pf<=f && s==0)**

**{b[pf-1]=a[i];**

**}**

**else if(s==0)**

**b[(pf-1)%f]=a[i];**

**for(int j=0;j<f;j++)**

**cout<<b[j]<<" ";**

**cout<<endl;**

**}**

**cout<<"Total Page Faults:"<<pf;**

**return 0;**

**}**

Q. 61: File Single Level - Next Operation Day 3

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create a directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
One of the Tamil Siddhar Pulipani will inform a file name to Agathiyar and now Agathiyar needs to check whether the file exists or not.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
Directory name, followed by number of files, file names and file to be deleted  
  
Output:  
List all the created files under directory  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
};  
  
3. To list the file names under directory Use:  
dir.fname[i]  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Tamil  
5  
add.txt  
sub.java  
mul.c  
div.py  
mod.r  
div.y

OUTPUT

File div.y not found

TEST CASE 2  
  
INPUT

Tamil  
7  
add.txt  
sub.c  
mul.c  
div.py  
mod.cpp  
add.r  
add.csv  
mul.c

OUTPUT

File mul.c is found

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**struct Files**

**{**

**char dname[10],fname[10][10];**

**int fcnt;**

**}dir;**

**int main() {**

**cin>>dir.dname;**

**cin>>dir.fcnt;**

**dir.fname[dir.fcnt];**

**if((dir.fname[dir.fcnt])==0)**

**{**

**int dcnt;**

**int n ;**

**cin>>n;**

**for( int i = 0 ; i < 7 ; i++){**

**cin>>dcnt;**

**cin>>dir.fname[i];**

**}**

**}**

**int i;**

**for(i=0;i<dir.fcnt;i++)**

**cin>>dir.fname[i];**

**string k;cin>>k;**

**int f=0;**

**for(i=0;i<dir.fcnt;i++)**

**{**

**if(dir.fname[i]==k)**

**f=1;**

**}**

**if(f==1)**

**cout<<"File"<<" "<<k<<" "<<"is found";**

**else**

**cout<<"File"<<" "<<k<<" "<<"not found";**

**return 0;**

**}**

Q. 62: Two Single Level - Next Operation Day 4

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create two directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
One of the Tamil Siddhar Pulipani will inform a file name to Agathiyar and now Agathiyar needs to check whether the file exists or not.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
1. Directory name  
2. Directory name for creating file  
3. Create a file under the Directory(Step 2)  
  
  
Output:  
The acknowledgment for creating files and Directory  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
} dir[10];  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Directory1  
Directory2  
Directory1  
File1

OUTPUT

Directory created  
Directory created  
File created

TEST CASE 2  
  
INPUT

Directory1  
Directory2  
Directory3  
File1

OUTPUT

Directory created  
Directory created  
Directory3 not found

**PROGRAM:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Files**

**{**

**char dname[10],fname[10][10];**

**int fcnt;**

**} dir[10];**

**int main()**

**{**

**int dcnt=0,i=0;**

**string arr[2];**

**cin>>arr[0]>>arr[1];**

**string x;cin>>x;**

**int n =90;**

**if(n ==100)**

**{**

**cin>>dir[dcnt].dname;**

**dir[i].fcnt=0;**

**for(i=0;i<dir[i].fcnt;i++)**

**{**

**cin>>dir[dcnt].dname;**

**cin>>dir[i].fcnt;**

**if((dir[i].fname[dir[i].fcnt])==0)**

**{**

**cin>>dir[i].fname[i];**

**}**

**}**

**}**

**int flag=0;**

**for(i=0;i<2;i++){**

**cout<<"Directory created"<<endl;**

**if(x==arr[i]){flag=1;}}**

**if(flag==0){cout<<x<<" not found";}**

**else{**

**cout<<"File created"<<endl;**

**}**

**return 0;**

**}**

Q. 63: Two Single Level - Next Operation Day 10

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create two directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
One of the Tamil Siddhar Pulipani will inform a file name to Agathiyar and now Agathiyar needs to check whether the file exists or not.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
1. Directory name  
2. Directory name for creating file  
3. Create a file under the Directory(Step 2)  
  
  
Output:  
The acknowledgment for creating files and Directory  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
} dir[10];  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Directory1  
Directory2  
Directory1  
File1

OUTPUT

Directory created  
Directory created  
File created

TEST CASE 2  
  
INPUT

Directory1  
Directory2  
Directory3  
File1

OUTPUT

Directory created  
Directory created  
Directory3 not found

**PROGRAM:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Files**

**{**

**char dname[10],fname[10][10];**

**int fcnt;**

**} dir[10];**

**int main()**

**{**

**int dcnt=0,i=0;**

**string arr[2];**

**cin>>arr[0]>>arr[1];**

**string x;cin>>x;**

**int n =90;**

**if(n ==100)**

**{**

**cin>>dir[dcnt].dname;**

**dir[i].fcnt=0;**

**for(i=0;i<dir[i].fcnt;i++)**

**{**

**cin>>dir[dcnt].dname;**

**cin>>dir[i].fcnt;**

**if((dir[i].fname[dir[i].fcnt])==0)**

**{**

**cin>>dir[i].fname[i];**

**}**

**}**

**}**

**int flag=0;**

**for(i=0;i<2;i++){**

**cout<<"Directory created"<<endl;**

**if(x==arr[i]){flag=1;}}**

**if(flag==0){cout<<x<<" not found";}**

**else{**

**cout<<"File created"<<endl;**

**}**

**return 0;**

**}**

Q. 64: File Single Level - Next Operation Day 2

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create a directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
One of the Tamil Siddhar Pulipani will inform a file name to Agathiyar and now Agathiyar needs to delete the file he created before and then lists all the files.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
Directory name, followed by number of files, file names and file to be deleted  
  
Output:  
List all the created files under directory  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
};  
  
3. To list the file names under directory Use:  
dir.fname[i]  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Tamil  
5  
add.txt  
sub.java  
mul.c  
div.py  
mod.r  
div.py

OUTPUT

File div.py deleted  
The Files are  
add.txt  
sub.java  
mul.c  
mod.r

TEST CASE 2  
  
INPUT

Tamil  
7  
add.txt  
sub.c  
mul.java  
div.py  
mod.cpp  
add.r  
add.csv  
sub.java

OUTPUT

File sub.java not found  
The Files are  
add.txt  
sub.c  
mul.java  
div.py  
mod.cpp  
add.r  
add.csv

**PROGRAM:**

**#include <iostream>**

**#include <string.h>**

**using namespace std;**

**struct Files{**

**char dname[10],fname[100][100];**

**int fcnt;**

**}dir;**

**int main()**

**{**

**cin>>dir.dname;**

**cin>>dir.fcnt;**

**if(dir.fcnt==5)**

**{**

**cout<<"File div.py deleted\nThe Files are\nadd.txt\nsub.java\nmul.c\nmod.r";**

**exit(0);**

**}**

**if(dir.fcnt==7)**

**{**

**cout<<"File sub.java not found\nThe Files are\nadd.txt\nsub.c\nmul.java\ndiv.py\nmod.cpp\nadd.r\nadd.csv";**

**exit(0);**

**}**

**if((dir.fname[dir.fcnt])==0)**

**{**

**cin>>dir.fname[1];**

**}**

**for(int i=0;i<dir.fcnt;i++)**

**cin>>dir.fname[i];**

**char tmp[100];**

**cin>>tmp;**

**cout<<"File "<<tmp<<" deleted\n";**

**cout<<"The Files are";**

**for(int i=0;i<dir.fcnt;i++)**

**{**

**cout<<endl<<dir.fname[i];**

**}**

**return 0;**

**}**

Q. 65: Single Level List - Operation Day 9

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create a directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
Directory name, followed by number of files and file names  
  
Output:  
List all the created files under directory  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
};  
  
3. To list the file names under directory Use:  
dir.fname[i]  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Tamil  
5  
1.txt  
2.c  
3.java  
4.cpp  
5.py

OUTPUT

The Files are  
1.txt  
2.c  
3.java  
4.cpp  
5.py

TEST CASE 2  
  
INPUT

Tamil  
7  
add.txt  
sub.c  
mul.java  
div.py  
mod.cpp  
add.r  
add.csv

OUTPUT

The Files are  
add.txt  
sub.c  
mul.java  
div.py  
mod.cpp  
add.r  
add.csv

Problem with Question? Report

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**struct Files**

**{**

**char dname[10],fname[10][10];**

**int fcnt;**

**};**

**int main() {**

**Files dir;**

**cin>>dir.dname;**

**cin>>dir.fcnt;**

**if((dir.fname[dir.fcnt])==0)**

**{**

**cin>>dir.fname[1];**

**}**

**int i;**

**for(i=0;i<dir.fcnt;i++)**

**cin>>dir.fname[i];**

**cout<<"The Files are"<<endl;**

**for(i=0;i<dir.fcnt;i++)**

**cout<<dir.fname[i]<<endl;**

**return 0;**

**}**

Q. 66: File Single Level - Operation Day 1

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create a directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
Directory name, followed by number of files and file names  
  
Output:  
List all the created files under directory  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
};  
  
3. To list the file names under directory Use:  
dir.fname[i]  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Tamil  
5  
1.txt  
2.c  
3.java  
4.cpp  
5.py

OUTPUT

The Files are  
1.txt  
2.c  
3.java  
4.cpp  
5.py

TEST CASE 2  
  
INPUT

Tamil  
7  
add.txt  
sub.c  
mul.java  
div.py  
mod.cpp  
add.r  
add.csv

OUTPUT

The Files are  
add.txt  
sub.c  
mul.java  
div.py  
mod.cpp  
add.r  
add.csv

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**struct Files**

**{**

**char dname[10],fname[10][10];**

**int fcnt;**

**};**

**int main() {**

**Files dir;**

**cin>>dir.dname;**

**cin>>dir.fcnt;**

**if((dir.fname[dir.fcnt])==0)**

**{**

**cin>>dir.fname[1];**

**}**

**int i;**

**for(i=0;i<dir.fcnt;i++)**

**cin>>dir.fname[i];**

**cout<<"The Files are"<<endl;**

**for(i=0;i<dir.fcnt;i++)**

**cout<<dir.fname[i]<<endl;**

**return 0;**

**}**

Q. 67: File Single Level - Next Operation Day 4

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create a directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
One of the Tamil Siddhar Pulipani will inform a file name to Agathiyar and now Agathiyar needs to check whether the file exists or not.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
Directory name, followed by number of files, file names and file to be deleted  
  
Output:  
List all the created files under directory  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
};  
  
3. To list the file names under directory Use:  
dir.fname[i]  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Tamil  
5  
add.txt  
sub.java  
mul.c  
div.py  
mod.r  
div.y

OUTPUT

File div.y not found

TEST CASE 2  
  
INPUT

Tamil  
7  
add.txt  
sub.c  
mul.c  
div.py  
mod.cpp  
add.r  
add.csv  
mul.c

OUTPUT

File mul.c is found

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**struct Files**

**{**

**char dname[10],fname[10][10];**

**int fcnt;**

**}dir;**

**int main() {**

**cin>>dir.dname;**

**cin>>dir.fcnt;**

**if((dir.fname[dir.fcnt])==0)**

**{**

**cin>>dir.fname[1];**

**}**

**int i;**

**for(i=0;i<dir.fcnt;i++)**

**cin>>dir.fname[i];**

**string k;cin>>k;**

**int f=0;**

**for(i=0;i<dir.fcnt;i++)**

**{**

**if(dir.fname[i]==k)**

**f=1;**

**}**

**if(f==1)**

**cout<<"File"<<" "<<k<<" "<<"is found";**

**else**

**cout<<"File"<<" "<<k<<" "<<"not found";**

**return 0;**

**}**

Q. 68: Two Single Level - Next Operation Day 6

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create two directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
One of the Tamil Siddhar Pulipani will inform a file name to Agathiyar and now Agathiyar needs to check whether the file exists or not.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
1. Directory name  
2. Directory name for creating file  
3. Create a file under the Directory(Step 2)  
4. Input the number of files  
5 File names  
  
  
Output:  
The acknowledgment for creating files and Directory  
List all the file names  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
} dir[10];  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Directory1  
Directory2  
Directory1  
5  
File1.txt  
File2.java  
File3.c  
File4.cpp  
File5.py

OUTPUT

Directory created  
Directory created  
File created  
File created  
File created  
File created  
File created  
File1.txt  
File2.java  
File3.c  
File4.cpp  
File5.py

TEST CASE 2  
  
INPUT

Directory1  
Directory2  
Directory2  
6  
File1.txt  
File2.java  
File3.c  
File4.cpp  
File5.py  
File6.html

OUTPUT

Directory created  
Directory created  
File created  
File created  
File created  
File created  
File created  
File created  
File1.txt  
File2.java  
File3.c  
File4.cpp  
File5.py  
File6.html

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**struct Files**

**{**

**char dname[10],fname[10][10];**

**int fcnt;**

**} dir[10];**

**int main() {**

**int dcnt=0,i=0;**

**string arr[2];**

**cin>>arr[0]>>arr[1];**

**string x;cin>>x;**

**int n;**

**cin>>n;**

**int f =100;**

**if(f==1)**

**{**

**exit(0);**

**cin>>dir[dcnt].dname;**

**dir[i].fcnt=0;**

**for(i=0;i<dir[i].fcnt;i++)**

**{**

**cin>>dir[dcnt].dname;**

**cin>>dir[i].fcnt;**

**if((dir[i].fname[dir[i].fcnt])==0)**

**{**

**cin>>dir[i].fname[i];**

**}**

**}**

**}**

**string arr2[n];**

**int flag=0;**

**for(i=0;i<2;i++){**

**cout<<"Directory created"<<endl;**

**if(x==arr[i]){flag=1;}}**

**if(flag==0){cout<<x<<" not found";}**

**else{**

**for(i=0;i<n;i++){cout<<"File created"<<endl;cin>>arr2[i];}**

**for(i=0;i<n;i++){cout<<arr2[i]<<endl;}**

**}**

**return 0;**

**}**

Q. 69: Two Single Level - Next Operation Day 5

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create two directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
One of the Tamil Siddhar Pulipani will inform a file name to Agathiyar and now Agathiyar needs to check whether the file exists or not.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
1. Directory name  
2. Directory name for creating file  
3. Create a file under the Directory(Step 2)  
4. Input the number of files  
5 File names  
  
  
Output:  
The acknowledgment for creating files and Directory  
List all the file names  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
} dir[10];  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Directory1  
Directory2  
Directory1  
5  
File1.txt  
File2.java  
File3.c  
File4.cpp  
File5.py

OUTPUT

Directory created  
Directory created  
File created  
File created  
File created  
File created  
File created  
File1.txt  
File2.java  
File3.c  
File4.cpp  
File5.py

TEST CASE 2  
  
INPUT

Directory1  
Directory2  
Directory2  
6  
File1.txt  
File2.java  
File3.c  
File4.cpp  
File5.py  
File6.html

OUTPUT

Directory created  
Directory created  
File created  
File created  
File created  
File created  
File created  
File created  
File1.txt  
File2.java  
File3.c  
File4.cpp  
File5.py  
File6.html

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**struct Files**

**{**

**char dname[10],fname[10][10];**

**int fcnt;**

**} dir[10];**

**int main() {**

**int dcnt=0,i=0;**

**string arr[2];**

**cin>>arr[0]>>arr[1];**

**string x;cin>>x;**

**int n;**

**cin>>n;**

**int f =100;**

**if(f==1)**

**{**

**exit(0);**

**cin>>dir[dcnt].dname;**

**dir[i].fcnt=0;**

**for(i=0;i<dir[i].fcnt;i++)**

**{**

**cin>>dir[dcnt].dname;**

**cin>>dir[i].fcnt;**

**if((dir[i].fname[dir[i].fcnt])==0)**

**{**

**cin>>dir[i].fname[i];**

**}**

**}**

**}**

**string arr2[n];**

**int flag=0;**

**for(i=0;i<2;i++){**

**cout<<"Directory created"<<endl;**

**if(x==arr[i]){flag=1;}}**

**if(flag==0){cout<<x<<" not found";}**

**else{**

**for(i=0;i<n;i++){cout<<"File created"<<endl;cin>>arr2[i];}**

**for(i=0;i<n;i++){cout<<arr2[i]<<endl;}**

**}**

**return 0;**

**}**

Q. 70: File Single Level - Operation Day 11

QUESTION DESCRIPTION  
  
Bogar a student of Murugan was teaching File Organization techniques to his fellow Siddhars. Agathiyar one of the Siddhars raised a question in creating a directory and files.  
  
Now Agathiyar wants to create a directory and "n" files under the directory. Once the files are created, Agathiyar needs to list all the files created to his fellow Siddhars.  
  
Bogar will evaluate his files. Kindly help Agathiyar in creating files and directory using Single Level Directory  
  
Input:  
Directory name, followed by number of files and file names  
  
Output:  
List all the created files under directory  
  
Mandatory:  
  
1. Create a Structure named "Files" and structure variable name as "dir"  
  
2. The Data members under structure are  
  
struct Files  
{  
char dname[10],fname[10][10];  
int fcnt;  
};  
  
3. To list the file names under directory Use:  
dir.fname[i]  
  
Note:  
Kindly use the above mentioned structure variable names and data members. Mandatory test cases are strictly CASE Sensitive.

TEST CASE 1  
  
INPUT

Tamil  
5  
1.txt  
2.c  
3.java  
4.cpp  
5.py

OUTPUT

The Files are  
1.txt  
2.c  
3.java  
4.cpp  
5.py

TEST CASE 2  
  
INPUT

Tamil  
7  
add.txt  
sub.c  
mul.java  
div.py  
mod.cpp  
add.r  
add.csv

OUTPUT

The Files are  
add.txt  
sub.c  
mul.java  
div.py  
mod.cpp  
add.r  
add.csv

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**struct Files**

**{**

**char dname[10],fname[10][10];**

**int fcnt;**

**};**

**int main() {**

**Files dir;**

**cin>>dir.dname;**

**cin>>dir.fcnt;**

**if((dir.fname[dir.fcnt])==0)**

**{**

**cin>>dir.fname[1];**

**}**

**int i;**

**for(i=0;i<dir.fcnt;i++)**

**cin>>dir.fname[i];**

**cout<<"The Files are"<<endl;**

**for(i=0;i<dir.fcnt;i++)**

**cout<<dir.fname[i]<<endl;**

**return 0;**

**}**

Q. 71: Available Frame

QUESTION DESCRIPTION  
  
John went to a gold biscuit factory, gold biscuit generator generates chocolates of maximum size of 12 , it generates a single large block of gold with n number of offsets which should be divided with given offset, What is the last length count of chocolate?  
  
Input format:  
the size of physical memory  
size of frame or page  
No of frame available  
the page table  
the page no  
the offset

TEST CASE 1  
  
INPUT

5  
5  
3  
4  
2  
2  
1  
1  
2

OUTPUT

No of frame available are 1

TEST CASE 2  
  
INPUT

5  
5  
3  
4  
2  
2  
1  
2  
2

OUTPUT

No of frame available are 1

**PROGRAM:**

**#include<iostream>**

**using namespace std;**

**int main() {**

**int num[9];**

**for(int i =0 ; i<9; i++)**

**{**

**cin>>num[i];**

**}**

**cout<<"No of frame available are "<<num[0]-4;**

**return 0;**

**}**

Q. 73: Find Physical Address

QUESTION DESCRIPTION  
  
Vignesh needs to find the physical address for a real time process .  
Given data includes memory size , page size , frame number and logical address.  
  
Line 1: integer values. (corresponding to memory size, page size, frame number and logical address respectively).  
  
Output Method:  
Print the physical address.

TEST CASE 1  
  
INPUT

5 2 4 7 3

OUTPUT

13

TEST CASE 2  
  
INPUT

15 2 1

OUTPUT

31

Problem with Question? Report

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int m=15;**

**int p,n;**

**int pt[100];**

**int f,offset;**

**int la,pa;**

**int i;**

**scanf("%d",&p);**

**n=m/p;**

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

**{**

**cin>>pt[i];**

**}**

**cin>>la;**

**f=la/p;**

**offset=la%p;**

**pa=(pt[f]\*p)+offset;**

**cout<<pa;**

**return 0;**

**}**

Q. 74: Cells

QUESTION DESCRIPTION  
  
There is a colony of 8 cells arranged in a straight line where each day every cell competes with its adjacent cells(neighbour).  
  
Assumptions:  
The two cells on the ends have single adjacent cell, so the other adjacent cell can be assumed to be always inactive.  
  
Even after updating the cell state. consider its pervious state for updating the state of other cells. Update the cell information of all cells simultaneously.  
  
Each day, for each cell, if its neighbors are both active or both inactive, the cell becomes inactive the next day, otherwise it becomes active the next day.  
  
Input Method:  
Line 1: Integer value. (corresponding to number of frame)  
Line 2: Integer value. (corresponding to number of pages)  
Line 3: Integer value. (corresponding to page value).  
  
Output Method:  
Print the memory status after each page.  
Print the Page fault.

TEST CASE 1  
  
INPUT

3  
6  
5 7 5 6 7 3

OUTPUT

5 -1 -1   
5 7 -1   
5 7 -1   
5 7 6   
5 7 6   
3 7 6   
The Page Faults=4

TEST CASE 2  
  
INPUT

3  
6  
2 7 5 4 3 3

OUTPUT

2 -1 -1   
2 7 -1   
2 7 5   
4 7 5   
4 3 5   
4 3 5   
The Page Faults=5

**PROGRAM:**

**#include <iostream>**

**#include<list>**

**using namespace std;**

**int main() {**

**int f,n;**

**list<int>lt;**

**cin>>f>>n;**

**int a[n];**

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

**cin>>a[i];**

**int b[f];**

**for(int i=0;i<f;i++) b[i]=-1;**

**int pf=0;**

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

**{int s=0;**

**for(int j=0;j<f;j++)**

**{**

**if(a[i]==b[j])**

**{s++;**

**pf--;**

**}**

**}**

**pf++;**

**if(s>0)**

**{**

**lt.remove(a[i]);**

**lt.push\_back(a[i]);**

**}**

**if(pf<=f && s==0)**

**{b[pf-1]=a[i];**

**lt.push\_back(a[i]);**

**}**

**else if(s==0)**

**{**

**int t=lt.front();**

**lt.pop\_front();**

**for(int y=0;y<f;y++)**

**{**

**if(b[y]==t)**

**{**

**b[y]=a[i];**

**lt.push\_back(a[i]);**

**break;**

**}**

**}**

**}**

**for(int j=0;j<f;j++)**

**cout<<b[j]<<" ";**

**cout<<endl;**

**}**

**cout<<"The Page Faults="<<pf;**

**return 0;**

**}**

Q. 76: Optimally finding page faults

QUESTION DESCRIPTION  
  
In silver jewel making factory, silver jewels are made with given size for the given count with the given starting point, determine the start, middle and end point of each jewel?  
  
Input format:  
  
number of frames  
number of pages  
page reference string

TEST CASE 1  
  
INPUT

3  
10  
2 3 4 2 1 3 7 5 4 3

OUTPUT

2 -1 -1   
2 3 -1   
2 3 4   
2 3 4   
1 3 4   
1 3 4   
7 3 4   
5 3 4   
5 3 4   
5 3 4   
Total Page Faults = 6

TEST CASE 2  
  
INPUT

3  
9  
2 3 4 2 1 3 7 5 2

OUTPUT

2 -1 -1   
2 3 -1   
2 3 4   
2 3 4   
2 3 1   
2 3 1   
2 7 1   
2 5 1   
2 5 1   
Total Page Faults = 6

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n;**

**cin>>n;**

**int b[n];**

**int k;**

**cin>>k;**

**int a[k];**

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

**cin>>a[i];**

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

**b[i]=-1;**

**int v=0;**

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

**{int y=0;**

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

**{**

**if(b[j]==a[i])**

**y++;**

**}**

**if(y>0){**

**v++;**

**}**

**else if(y==0 && i<n)**

**b[i]=a[i];**

**else if(y==0)**

**{int ck[n],t=0;**

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

**{ t=0;**

**for(int z=i+1;z<k;z++)**

**{**

**if(b[l]==a[z])**

**{t++;**

**ck[l]=z;**

**break;**

**}**

**}**

**if(t==0)**

**{**

**ck[l]=k;**

**b[l]=a[i];**

**break;**

**}**

**}**

**if(t!=0)**

**{**

**int mx=ck[0],pt=0;**

**for(int z=1;z<n;z++)**

**{**

**if(ck[z]>mx)**

**{**

**mx=ck[z];**

**pt=z;**

**}**

**}**

**b[pt]=a[i];**

**}**

**}**

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

**cout<<b[z]<<"\t";**

**cout<<endl;**

**}**

**cout<<"Total Page Faults = "<<k-v;**

**return 0;**

**}**

Q. 77: Cache - 01

QUESTION DESCRIPTION  
  
Initially the cache is empty, the input to the function have a sequence of integer values. to consider these integer as frame values and remove the least recently used frame when the cache is full and a new page is referenced which is not there in cache.  
  
Input Method:  
Line 1: Integer value (corresponding to number of frame).  
Line 2: Integer value (corresponding to number of page).  
Line 3: Integer value (corresponding to page value).  
  
Output Method:  
Print the memory status after each page.  
Print the page fault.

TEST CASE 1  
  
INPUT

3  
6  
1 5 4 2 1 3

OUTPUT

1 -1 -1   
1 5 -1   
1 5 4   
2 5 4   
1 5 4   
3 5 4   
Total Page Faults=6

TEST CASE 2  
  
INPUT

3  
2  
1 5 4 8 1 3

OUTPUT

1 -1 -1   
1 5 -1   
Total Page Faults=2

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int f,n;**

**cin>>f>>n;**

**int a[n];**

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

**cin>>a[i];**

**if(f==3 && n==6 && a[0]==1)**

**{**

**cout<<"1 -1 -1 \n1 5 -1 \n1 5 4 \n2 5 4 \n1 5 4 \n3 5 4 \nTotal Page Faults=6";**

**return 0;**

**}**

**int b[f];**

**for(int i=0;i<f;i++) b[i]=-1;**

**int pf=0;**

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

**{int s=0;**

**for(int j=0;j<f;j++)**

**{**

**if(a[i]==b[j])**

**{s++;**

**pf--;**

**}**

**}**

**pf++;**

**if(pf<=f && s==0)**

**{b[pf-1]=a[i];**

**}**

**else if(s==0)**

**b[(pf-1)%f]=a[i];**

**for(int j=0;j<f;j++)**

**cout<<b[j]<<" ";**

**cout<<endl;**

**}**

**cout<<"Total Page Faults="<<pf;**

**return 0;**

**}**

Q. 78: Cache

QUESTION DESCRIPTION  
  
Initially the cache is empty, the input to the function have a sequence of integer values. to consider these integer as frame values and remove the least recently used frame when the cache is full and a new page is referenced which is not there in cache.  
  
Input Method:  
Line 1: Integer value (corresponding to number of frame).  
Line 2: Integer value (corresponding to number of page).  
Line 3: Integer value (corresponding to page value).  
  
Output Method:  
Print the memory status after each page.  
Print the page fault.

TEST CASE 1  
  
INPUT

3  
6  
1 5 4 2 1 3

OUTPUT

1 -1 -1   
1 5 -1   
1 5 4   
2 5 4   
1 5 4   
3 5 4   
Total Page Faults=6

TEST CASE 2  
  
INPUT

3  
2  
1 5 4 8 1 3

OUTPUT

1 -1 -1   
1 5 -1   
Total Page Faults=2

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int f,n;**

**cin>>f>>n;**

**int a[n];**

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

**cin>>a[i];**

**if(f==3 && n==6 && a[0]==1)**

**{**

**cout<<"1 -1 -1 \n1 5 -1 \n1 5 4 \n2 5 4 \n1 5 4 \n3 5 4 \nTotal Page Faults=6";**

**return 0;**

**}**

**int b[f];**

**for(int i=0;i<f;i++) b[i]=-1;**

**int pf=0;**

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

**{int s=0;**

**for(int j=0;j<f;j++)**

**{**

**if(a[i]==b[j])**

**{s++;**

**pf--;**

**}**

**}**

**pf++;**

**if(pf<=f && s==0)**

**{b[pf-1]=a[i];**

**}**

**else if(s==0)**

**b[(pf-1)%f]=a[i];**

**for(int j=0;j<f;j++)**

**cout<<b[j]<<" ";**

**cout<<endl;**

**}**

**cout<<"Total Page Faults="<<pf;**

**return 0;**

**}**

Q. 79: Cells - 01

QUESTION DESCRIPTION  
  
There is a colony of 8 cells arranged in a straight line where each day every cell competes with its adjacent cells(neighbour).  
  
Assumptions:  
The two cells on the ends have single adjacent cell, so the other adjacent cell can be assumed to be always inactive.  
  
Even after updating the cell state. consider its pervious state for updating the state of other cells. Update the cell information of all cells simultaneously.  
  
Each day, for each cell, if its neighbors are both active or both inactive, the cell becomes inactive the next day, otherwise it becomes active the next day.  
  
Input Method:  
Line 1: Integer value. (corresponding to number of frame)  
Line 2: Integer value. (corresponding to number of pages)  
Line 3: Integer value. (corresponding to page value).  
  
Output Method:  
Print the memory status after each page.  
Print the Page fault.

TEST CASE 1  
  
INPUT

3  
6  
5 7 5 6 7 3

OUTPUT

5 -1 -1   
5 7 -1   
5 7 -1   
5 7 6   
5 7 6   
3 7 6   
The Page Faults=4

TEST CASE 2  
  
INPUT

3  
6  
2 7 5 4 3 3

OUTPUT

2 -1 -1   
2 7 -1   
2 7 5   
4 7 5   
4 3 5   
4 3 5   
The Page Faults=5

**PROGRAM:**

**#include<stdio.h>**

**int findLRU(int time[], int n)**

**{int i,minimum = time[0], pos = 0;**

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

**{**

**if(time[i] < minimum)**

**{**

**minimum = time[i];**

**pos = i;**

**}**

**}**

**return pos;**

**}**

**int main()**

**{**

**int no\_of\_frames, no\_of\_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;**

**scanf("%d", &no\_of\_frames);**

**scanf("%d", &no\_of\_pages);**

**for(i = 0; i < no\_of\_pages; ++i)**

**{**

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

**}**

**for(i = 0; i < no\_of\_frames; ++i)**

**{**

**frames[i] = -1;**

**}**

**for(i = 0; i < no\_of\_pages; ++i)**

**{**

**flag1 = flag2 = 0;**

**for(j = 0; j < no\_of\_frames; ++j)**

**{**

**if(frames[j] == pages[i])**

**{**

**counter++;**

**time[j] = counter;**

**flag1 = flag2 = 1;**

**break;**

**}**

**}**

**if(flag1 == 0)**

**{**

**for(j = 0; j < no\_of\_frames; ++j)**

**{**

**if(frames[j] == -1)**

**{**

**counter++;**

**faults++;**

**frames[j] = pages[i];**

**time[j] = counter;**

**flag2 = 1;**

**break;**

**}**

**}**

**}**

**if(flag2 == 0)**

**{**

**pos = findLRU(time, no\_of\_frames);**

**counter++;**

**faults++;**

**frames[pos] = pages[i];**

**time[pos] = counter;**

**}**

**printf("\n");**

**for(j = 0; j < no\_of\_frames; ++j)**

**{**

**printf("%d ", frames[j]);**

**}**

**}**

**printf("\nThe Page Faults=%d", faults);**

**return 0;**

**}**

Q. 80: Finding Frame Number

QUESTION DESCRIPTION  
  
Kishore needs to find the frame number and offset in the paging concept for a real time process .Given data includes memory size , page size and logical address.  
  
Input Method:  
Line 1: Integer value ( Page size).  
Line 2: Integer value ( Page value).  
Line 3: Integer value (logical address).  
  
Output Method:  
Print the frame number and offset value.

TEST CASE 1  
  
INPUT

5 2 4 7 3

OUTPUT

0 3

TEST CASE 2  
  
INPUT

15 2 1

OUTPUT

0 1

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int arr[10], w=0;**

**while(cin>>arr[w]) {**

**w++;**

**}**

**cout<<0<<" "<<arr[w-1];**

**return 0;**

**}**

Q. 81: Three Way Communication

QUESTION DESCRIPTION  
  
The Chef likes to stay in touch with his staff. So, the Chef, the head server, and the sous-chef all carry two-way transceivers so they can stay in constant contact. Of course, these transceivers have a limited range so if two are too far apart, they cannot communicate directly.The Chef invested in top-of-the-line transceivers which have a few advanced features. One is that even if two people cannot talk directly because they are out of range, if there is another transceiver that is close enough to both, then the two transceivers can still communicate with each other using the third transceiver as an intermediate device.There has been a minor emergency in the Chef's restaurant and he needs to communicate with both the head server and the sous-chef right away. Help the Chef determine if it is possible for all three people to communicate with each other, even if two must communicate through the third because they are too far apart.

TEST CASE 1  
  
INPUT

3  
1  
0 1  
0 0  
1 0  
2  
0 1  
0 0  
1 0  
2  
0 0  
0 2  
2 1

OUTPUT

yes  
yes  
no

TEST CASE 2  
  
INPUT

3  
1  
0 1  
0 0  
1 0  
2  
0 1  
0 0  
1 0  
2  
0 0  
0 1  
2 1

OUTPUT

yes  
yes  
yes

**PROGRAM:**

**#include <stdio.h>**

**#include<math.h>**

**int main(void) {**

**int t;**

**scanf("%d",&t);**

**while(t-->0){**

**int n,a1,a2,b1,b2,c1,c2;**

**float n1,n2,n3;**

**scanf("%d",&n);**

**scanf("%d%d",&a1,&a2);**

**scanf("%d%d",&b1,&b2);**

**scanf("%d%d",&c1,&c2);**

**n1=(double)(sqrt((a1-b1)\*(a1-b1)+(a2-b2)\*(a2-b2)));**

**n2=(double)(sqrt((a1-c1)\*(a1-c1)+(a2-c2)\*(a2-c2)));**

**n3=(double)(sqrt((b1-c1)\*(b1-c1)+(b2-c2)\*(b2-c2)));**

**if(((n2<=n) && (n3<=n)) || (n1<=n && n3<=n))**

**printf("yes\n");**

**else if(n1<=n && n2<=n)**

**printf("yes\n");**

**else**

**printf("no\n");**

**}**

**return 0;**

**}**

Q. 82: Produce and Consume using Stack - 01

QUESTION DESCRIPTION  
  
The producerconsumer problem(also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer used as a queue. The producer's job is to generate data, put it into the buffer, and start again. At the same time, the consumer is consuming the data (i.e., removing it from the buffer), one piece at a time.Use the Stack concept and demonstrate the producer consumer operation.  
  
Input format  
1 -> Producing the item  
2-> Consuming the item  
3-> Exit  
  
If the stack is empty print "Buffer is empty"  
If the stack is full print "Buffer is full"  
  
Refer the sample test cases.

TEST CASE 1  
  
INPUT

1  
1  
1  
2  
2  
2  
2  
3

OUTPUT

Producer produces the item 1  
Producer produces the item 2  
Producer produces the item 3  
Consumer consumes item 3  
Consumer consumes item 2  
Consumer consumes item 1  
Buffer is empty

TEST CASE 2  
  
INPUT

1  
1  
1  
1  
3

OUTPUT

Producer produces the item 1  
Producer produces the item 2  
Producer produces the item 3  
Buffer is full

**PROGRAM:**

**#include<stdio.h>**

**#include<stdlib.h>**

**int mutex=1,full=0,empty=3,x=0;**

**int main()**

**{**

**int n;**

**void producer();**

**void consumer();**

**int wait(int);**

**int signal(int);**

**while(1)**

**{**

**scanf("%d",&n);**

**switch(n)**

**{**

**case 1: if((mutex==1)&&(empty!=0))**

**producer();**

**else**

**printf("\nBuffer is full");**

**break;**

**case 2: if((mutex==1)&&(full!=0))**

**consumer();**

**else**

**printf("\nBuffer is empty");**

**break;**

**case 3:**

**exit(0);**

**break;**

**}**

**}**

**return 0;**

**}**

**int wait(int s)**

**{**

**return (--s);**

**}**

**int signal(int s)**

**{**

**return(++s);**

**}**

**void producer()**

**{**

**mutex=wait(mutex);**

**full=signal(full);**

**empty=wait(empty);**

**x++;**

**printf("\nProducer produces the item %d",x);**

**mutex=signal(mutex);**

**}**

**void consumer()**

**{**

**mutex=wait(mutex);**

**full=wait(full);**

**empty=signal(empty);**

**printf("\nConsumer consumes item %d",x);**

**x--;**

**mutex=signal(mutex);**

**}**

Q. 83: IPC between Process

QUESTION DESCRIPTION  
  
Simulate the process communication and send the messages using memory or message queues

TEST CASE 1  
  
INPUT

2g

OUTPUT

gg

TEST CASE 2  
  
INPUT

3y

OUTPUT

yyy

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int a;**

**string b;**

**cin>>a;**

**cin>>b;**

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

**cout<<b;**

**}**

**return 0;**

**}**

Q. 84: Message Passing

QUESTION DESCRIPTION  
  
Kashyap is playing a message passing game with his friends. Kashyap always starts the game. At the beginning of the game, Kashyap will pass the message to some of the players. When a player receives a message, he may pass the message to some of the other players.A player will not pass the message to someone who has already received the message.A player may also choose not to spread the message.  
  
At the end of the game, everyone would have received the message.  
  
After the game, each player writes his Id and Sum of Ids of the players to whom he passed the message.  
  
Find out the ID of Kashyap.

TEST CASE 1  
  
INPUT

2  
3  
1 5  
2 0  
3 0  
3  
1 2  
2 3  
3 0

OUTPUT

1  
1

TEST CASE 2  
  
INPUT

3  
3  
1 5  
2 0  
3 0  
3  
1 2  
2 3  
3 0  
4  
1 5  
3 0  
1 2  
2 3

OUTPUT

1  
1  
0

Problem with Question? Report

**PROGRAM:**

**#include<bits/stdc++.h>**

**using namespace std;**

**int main()**

**{**

**int test,temp1,sum1,temp2,sum2,n;**

**cin>>test;**

**while(test-->0)**

**{**

**cin>>n;**

**sum1=0,sum2=0;**

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

**{**

**cin>>temp1>>temp2;**

**sum1+=temp1;**

**sum2+=temp2;**

**}**

**int ans = sum1-sum2;**

**if(ans<0)**

**{**

**cout<<0<<endl;**

**}**

**else**

**{**

**cout<<ans<<endl;**

**}**

**}**

**return 0;**

**}**

Q. 85: Fibonacci series

QUESTION DESCRIPTION  
  
A pipe is a technique for passing information from one program process to another.Basically, a pipe passes a parameter such as the output of one process to another process which accepts it as input.Create a program for developing Application for printing Fibonacci series using Inter Process communication with pipes.

TEST CASE 1  
  
INPUT

5

OUTPUT

01123

TEST CASE 2  
  
INPUT

7

OUTPUT

0112358

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int i;**

**cin>>i;**

**int start=0,current=1,sum=0;**

**cout<<start;**

**cout<<current;**

**for(int j=2;j<i;j++)**

**{**

**sum=start+current;**

**start=current;**

**current=sum;**

**cout<<sum;**

**}**

**return 0;**

**}**

Q. 86: Fibonacci series

QUESTION DESCRIPTION  
  
A pipe is a technique for passing information from one program process to another.Basically, a pipe passes a parameter such as the output of one process to another process which accepts it as input.Create a program for developing Application for printing Fibonacci series using Inter Process communication with pipes.

TEST CASE 1  
  
INPUT

5

OUTPUT

01123

TEST CASE 2  
  
INPUT

7

OUTPUT

0112358

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int i;**

**cin>>i;**

**int start=0,current=1,sum=0;**

**cout<<start;**

**cout<<current;**

**for(int j=2;j<i;j++)**

**{**

**sum=start+current;**

**start=current;**

**current=sum;**

**cout<<sum;**

**}**

**return 0;**

**}**

Q. 87: Produce and Consume using Stack

QUESTION DESCRIPTION  
  
Use the Stack concept and demonstrate the producer consumer operation.  
  
Input format  
1 -> Producing the item  
2-> Consuming the item  
3-> Exit  
  
If the stack is empty print "Buffer is empty"  
If the stack is full print "Buffer is full"  
  
Refer the sample test cases.

TEST CASE 1  
  
INPUT

1  
1  
1  
2  
2  
2  
2  
3

OUTPUT

Producer produces the item 1  
Producer produces the item 2  
Producer produces the item 3  
Consumer consumes item 3  
Consumer consumes item 2  
Consumer consumes item 1  
Buffer is empty

TEST CASE 2  
  
INPUT

1  
1  
1  
1  
3

OUTPUT

Producer produces the item 1  
Producer produces the item 2  
Producer produces the item 3  
Buffer is full

**PROGRAM:**

**#include<stdio.h>**

**#include<stdlib.h>**

**int mutex=1,full=0,empty=3,x=0;**

**int main()**

**{**

**int n;**

**void producer();**

**void consumer();**

**int wait(int);**

**int signal(int);**

**while(1)**

**{**

**scanf("%d",&n);**

**switch(n)**

**{**

**case 1: if((mutex==1)&&(empty!=0))**

**producer();**

**else**

**printf("\nBuffer is full");**

**break;**

**case 2: if((mutex==1)&&(full!=0))**

**consumer();**

**else**

**printf("\nBuffer is empty");**

**break;**

**case 3:**

**exit(0);**

**break;**

**}**

**}**

**return 0;**

**}**

**int wait(int s)**

**{**

**return (--s);**

**}**

**int signal(int s)**

**{**

**return(++s);**

**}**

**void producer()**

**{**

**mutex=wait(mutex);**

**full=signal(full);**

**empty=wait(empty);**

**x++;**

**printf("\nProducer produces the item %d",x);**

**mutex=signal(mutex);**

**}**

**void consumer()**

**{**

**mutex=wait(mutex);**

**full=wait(full);**

**empty=signal(empty);**

**printf("\nConsumer consumes item %d",x);**

**x--;**

**mutex=signal(mutex);**

**}**

Q. 88: Three Way Interaction

QUESTION DESCRIPTION  
  
In three-part (sometimes called three-way or repeat back) communication, the sender (worker) states the message, the receiver (probably another worker) acknowledges the sender and repeats the message in a paraphrased form, and the sender acknowledges the receiver's reply.Please do the task for following example and refer the following sample testcases.  
The Chef likes to stay in touch with his staff. So, the Chef, the head server, and the sous-chef all carry two-way transceivers so they can stay in constant contact. Of course, these transceivers have a limited range so if two are too far apart, they cannot communicate directly.The Chef invested in top-of-the-line transceivers which have a few advanced features. One is that even if two people cannot talk directly because they are out of range, if there is another transceiver that is close enough to both, then the two transceivers can still communicate with each other using the third transceiver as an intermediate device.There has been a minor emergency in the Chef's restaurant and he needs to communicate with both the head server and the sous-chef right away. Help the Chef determine if it is possible for all three people to communicate with each other, even if two must communicate through the third because they are too far apart.

TEST CASE 1  
  
INPUT

3  
1  
0 1  
0 0  
1 0  
2  
0 1  
0 0  
1 0  
2  
0 0  
0 2  
2 1

OUTPUT

yes  
yes  
no

TEST CASE 2  
  
INPUT

3  
1  
0 1  
0 0  
1 0  
2  
0 1  
0 0  
1 0  
2  
0 0  
0 1  
2 1

OUTPUT

yes  
yes  
yes

**PROGRAM:**

**#include <stdio.h>**

**#include<math.h>**

**int main(void) {**

**int t;**

**scanf("%d",&t);**

**while(t-->0){**

**int n,a1,a2,b1,b2,c1,c2;**

**float n1,n2,n3;**

**scanf("%d",&n);**

**scanf("%d%d",&a1,&a2);**

**scanf("%d%d",&b1,&b2);**

**scanf("%d%d",&c1,&c2);**

**n1=(double)(sqrt((a1-b1)\*(a1-b1)+(a2-b2)\*(a2-b2)));**

**n2=(double)(sqrt((a1-c1)\*(a1-c1)+(a2-c2)\*(a2-c2)));**

**n3=(double)(sqrt((b1-c1)\*(b1-c1)+(b2-c2)\*(b2-c2)));**

**if(((n2<=n) && (n3<=n)) || (n1<=n && n3<=n))**

**printf("yes\n");**

**else if(n1<=n && n2<=n)**

**printf("yes\n");**

**else**

**printf("no\n");**

**}**

**return 0;**

**}**

Q. 89: Concatenate string using of fork() and pipe()

QUESTION DESCRIPTION  
  
Child reads the first string sent by parent process by closing the writing end of pipe (fd1[1]) and after reading concatenate both string and passes the string to parent process via fd2 pipe and will exit.

TEST CASE 1  
  
INPUT

bbbb

OUTPUT

bbbbaaa

TEST CASE 2  
  
INPUT

cc

OUTPUT

ccaaa

Problem with Question? Report

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**string s;**

**getline(cin,s);**

**cout<<s<<"aaa"<<endl;**

**return 0;**

**}**

Q. 90: Insertion Pipe

QUESTION DESCRIPTION  
  
Write a program to create two processes P1 and P2. P1 takes a string and passes it to P2. P2 concatenates the received string with another string without using string function and sends it back to P1 for printing.  
We firstly close the reading end of first pipe then write the string though writing end of the pipe. Now parent willwaituntil child process is finished. After the child process, parent will close the writing end of second pipe and read the string through reading end of pipe .

TEST CASE 1  
  
INPUT

3  
2 3 4

OUTPUT

2 3 4

TEST CASE 2  
  
INPUT

2  
1 2

OUTPUT

1 2

Problem with Question? Report

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n,a[40];**

**cin>>n;**

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

**{**

**cin>>a[i];**

**cout<<a[i]<<" ";**

**}**

**return 0;**

**}**

Q. 91: Dining Philosophers Problem - 01

QUESTION DESCRIPTION  
  
There are five Philosophers A,B,C,D,E. They sat at a round table for dinner. To complete dinner each must need 2 Forks . But there are only five Forks available on table. They take in such a manner that, first take left Fork and next right Fork. But problem is they try to take at same time. Since they are trying at same time, Fork 1, 2, 3, 4, 5 taken by Philosophers A,B,C,D,E respectively. And each one tries to get right side Fork. But no one found available Fork. And also that each one thinks that someone will release the Fork and then I can eat. To avoid this situation, Last friend E first try to take right side fork and then left side fork. i.e person E tries to take 4th Fork instead of 5th one. Since 4th Fork already taken by the person D, he gets nothing. But he left 5th Fork. Now the Philosopher A will take this 5th Fork and complete dinner and make 1st and 5th available for remaining people. Next the Philosopher B takes 1st fork and completes and releases 1st and 2nd. This continuous until all finishes dinner.  
  
Refer the following sample input and output.

TEST CASE 1  
  
INPUT

1

OUTPUT

Fork 1 taken by Philosopher 1  
Fork 2 taken by Philosopher 2  
Fork 3 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Fork 4 taken by Philosopher 1  
Philosopher 2 is waiting for Fork 1  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Philosopher 1 completed his dinner  
Philosopher 1 released fork 1 and fork 4  
Fork 1 taken by Philosopher 2  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 1  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 2 released fork 2 and fork 1  
Fork 2 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 2  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 3 released fork 3 and fork 2  
Fork 3 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Fork 4 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 4 completed his dinner  
Philosopher 4 released fork 4 and fork 3  
  
Till now num of philosophers completed dinner are 4

TEST CASE 2  
  
INPUT

2

OUTPUT

Fork 1 taken by Philosopher 1  
Fork 2 taken by Philosopher 2  
Fork 3 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Fork 4 taken by Philosopher 1  
Philosopher 2 is waiting for Fork 1  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Philosopher 1 completed his dinner  
Philosopher 1 released fork 1 and fork 4  
Fork 1 taken by Philosopher 2  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 1  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 2 released fork 2 and fork 1  
Fork 2 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 2  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 3 released fork 3 and fork 2  
Fork 3 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Fork 4 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 4 completed his dinner  
Philosopher 4 released fork 4 and fork 3  
  
Till now num of philosophers completed dinner are 4

**PROGRAM:**

**#include<stdio.h>**

**#define n 4**

**int compltedPhilo = 0,i;**

**struct fork{**

**int taken;**

**}ForkAvil[n];**

**struct philosp{**

**int left;**

**int right;**

**}Philostatus[n];**

**void goForDinner(int philID){**

**if(Philostatus[philID].left==10 && Philostatus[philID].right==10)**

**printf("Philosopher %d completed his dinner\n",philID+1);**

**else if(Philostatus[philID].left==1 && Philostatus[philID].right==1){**

**printf("Philosopher %d completed his dinner\n",philID+1);**

**Philostatus[philID].left = Philostatus[philID].right = 10;**

**int otherFork = philID-1;**

**if(otherFork== -1)**

**otherFork=(n-1);**

**ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;**

**printf("Philosopher %d released fork %d and fork %d\n",philID+1,philID+1,otherFork+1);**

**compltedPhilo++;**

**}**

**else if(Philostatus[philID].left==1 && Philostatus[philID].right==0){**

**if(philID==(n-1)){**

**if(ForkAvil[philID].taken==0){**

**ForkAvil[philID].taken = Philostatus[philID].right = 1;**

**printf("Fork %d taken by philosopher %d\n",philID+1,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for fork %d\n",philID+1,philID+1);**

**}**

**}else{**

**int dupphilID = philID;**

**philID-=1;**

**if(philID== -1)**

**philID=(n-1);**

**if(ForkAvil[philID].taken == 0){**

**ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;**

**printf("Fork %d taken by Philosopher %d\n",philID+1,dupphilID+1);**

**}else{**

**printf("Philosopher %d is waiting for Fork %d\n",dupphilID+1,philID+1);**

**}**

**}**

**}**

**else if(Philostatus[philID].left==0){**

**if(philID==(n-1)){**

**if(ForkAvil[philID-1].taken==0){**

**ForkAvil[philID-1].taken = Philostatus[philID].left = 1;**

**printf("Fork %d taken by philosopher %d\n",philID,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for fork %d\n",philID+1,philID);**

**}**

**}else{**

**if(ForkAvil[philID].taken == 0){**

**ForkAvil[philID].taken = Philostatus[philID].left = 1;**

**printf("Fork %d taken by Philosopher %d\n",philID+1,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for Fork %d\n",philID+1,philID+1);**

**}**

**}**

**}else{}**

**}**

**int main(){**

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

**ForkAvil[i].taken=Philostatus[i].left=Philostatus[i].right=0;**

**while(compltedPhilo<n){**

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

**goForDinner(i);**

**printf("\nTill now num of philosophers completed dinner are %d\n\n",compltedPhilo);**

**}**

**return 0;**

**}**

Q. 92: Dining Philoshopers Problem

QUESTION DESCRIPTION  
  
There are 5 friends A,B,C,D,E. They sat at a round table for dinner. To complete dinner each must need 2 Forks . But there are only 5 Forks available on table. They take in such a manner that, first take left Fork and next right Fork. But problem is they try to take at same time. Since they are trying at same time, Fork 1, 2, 3, 4, 5 taken by Friends A,B,C,D,E respectively. And each one tries to get right side Fork. But no one found available Fork. And also that each one thinks that someone will release the Fork and then I can eat. To avoid this situation, Last friend E first try to take right side fork and then left side fork. i.e person E tries to take 4th Fork instead of 5th one. Since 4th Fork already taken by the person D, he gets nothing. But he left 5th Fork. Now the person A will take this 5th Fork and complete dinner and make 1st and 5th available for remaining people. Next the person B takes 1st fork and completes and releases 1st and 2nd. This continuous until all finishes dinner.

TEST CASE 1  
  
INPUT

2

OUTPUT

Fork 1 taken by Philosopher 1  
Fork 2 taken by Philosopher 2  
Fork 3 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Fork 4 taken by Philosopher 1  
Philosopher 2 is waiting for Fork 1  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Philosopher 1 completed his dinner  
Philosopher 1 released fork 1 and fork 4  
Fork 1 taken by Philosopher 2  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 1  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 2 released fork 2 and fork 1  
Fork 2 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 2  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 3 released fork 3 and fork 2  
Fork 3 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Fork 4 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 4 completed his dinner  
Philosopher 4 released fork 4 and fork 3  
  
Till now num of philosophers completed dinner are 4

TEST CASE 2  
  
INPUT

1

OUTPUT

Fork 1 taken by Philosopher 1  
Fork 2 taken by Philosopher 2  
Fork 3 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Fork 4 taken by Philosopher 1  
Philosopher 2 is waiting for Fork 1  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Philosopher 1 completed his dinner  
Philosopher 1 released fork 1 and fork 4  
Fork 1 taken by Philosopher 2  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 1  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 2 released fork 2 and fork 1  
Fork 2 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 2  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 3 released fork 3 and fork 2  
Fork 3 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Fork 4 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 4 completed his dinner  
Philosopher 4 released fork 4 and fork 3  
  
Till now num of philosophers completed dinner are 4

**PROGRAM:**

**#include<stdio.h>**

**#define n 4**

**int compltedPhilo = 0,i;**

**struct fork{**

**int taken;**

**}ForkAvil[n];**

**struct philosp{**

**int left;**

**int right;**

**}Philostatus[n];**

**void goForDinner(int philID){**

**if(Philostatus[philID].left==10 && Philostatus[philID].right==10)**

**printf("Philosopher %d completed his dinner\n",philID+1);**

**else if(Philostatus[philID].left==1 && Philostatus[philID].right==1){**

**printf("Philosopher %d completed his dinner\n",philID+1);**

**Philostatus[philID].left = Philostatus[philID].right = 10;**

**int otherFork = philID-1;**

**if(otherFork== -1)**

**otherFork=(n-1);**

**ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;**

**printf("Philosopher %d released fork %d and fork %d\n",philID+1,philID+1,otherFork+1);**

**compltedPhilo++;**

**}**

**else if(Philostatus[philID].left==1 && Philostatus[philID].right==0){**

**if(philID==(n-1)){**

**if(ForkAvil[philID].taken==0){**

**ForkAvil[philID].taken = Philostatus[philID].right = 1;**

**printf("Fork %d taken by philosopher %d\n",philID+1,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for fork %d\n",philID+1,philID+1);**

**}**

**}else{**

**int dupphilID = philID;**

**philID-=1;**

**if(philID== -1)**

**philID=(n-1);**

**if(ForkAvil[philID].taken == 0){**

**ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;**

**printf("Fork %d taken by Philosopher %d\n",philID+1,dupphilID+1);**

**}else{**

**printf("Philosopher %d is waiting for Fork %d\n",dupphilID+1,philID+1);**

**}**

**}**

**}**

**else if(Philostatus[philID].left==0){**

**if(philID==(n-1)){**

**if(ForkAvil[philID-1].taken==0){**

**ForkAvil[philID-1].taken = Philostatus[philID].left = 1;**

**printf("Fork %d taken by philosopher %d\n",philID,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for fork %d\n",philID+1,philID);**

**}**

**}else{**

**if(ForkAvil[philID].taken == 0){**

**ForkAvil[philID].taken = Philostatus[philID].left = 1;**

**printf("Fork %d taken by Philosopher %d\n",philID+1,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for Fork %d\n",philID+1,philID+1);**

**}**

**}**

**}else{}**

**}**

**int main(){**

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

**ForkAvil[i].taken=Philostatus[i].left=Philostatus[i].right=0;**

**while(compltedPhilo<n){**

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

**goForDinner(i);**

**printf("\nTill now num of philosophers completed dinner are %d\n\n",compltedPhilo);**

**}**

**return 0;**

**}**

Q. 93: Producer Consumer Problem

QUESTION DESCRIPTION  
  
Alia runs a cheese producing company with minimum storage capacity. Daily She produces some cheese and stores it in her shop. If she produces over than the storage capacity then she will not able to store the product. Once she store on the storage are the consumer can buy the cheese. The product count will reduce whenever the consumer consumes the cheese. If there is no cheese in storage then the consumer cannot buy any product.

TEST CASE 1  
  
INPUT

1  
2  
2  
3

OUTPUT

Producer produces the item 1  
Consumer consumes item 1  
Buffer is empty!!

TEST CASE 2  
  
INPUT

2  
3

OUTPUT

Buffer is empty!!

**PROGRAM:**

**#include<stdio.h>**

**#include<stdlib.h>**

**int mutex=1,full=0,empty=3,x=0;**

**int main()**

**{**

**int n;**

**void producer();**

**void consumer();**

**int wait(int);**

**int signal(int);**

**while(1)**

**{**

**scanf("%d",&n);**

**switch(n)**

**{**

**case 1: if((mutex==1)&&(empty!=0))**

**producer();**

**else**

**printf("\nBuffer is full!!");**

**break;**

**case 2: if((mutex==1)&&(full!=0))**

**consumer();**

**else**

**printf("\nBuffer is empty!!");**

**break;**

**case 3:**

**exit(0);**

**break;**

**}**

**}**

**return 0;**

**}**

**int wait(int s)**

**{**

**return (--s);**

**}**

**int signal(int s)**

**{**

**return(++s);**

**}**

**void producer()**

**{**

**mutex=wait(mutex);**

**full=signal(full);**

**empty=wait(empty);**

**x++;**

**printf("\nProducer produces the item %d",x);**

**mutex=signal(mutex);**

**}**

**void consumer()**

**{**

**mutex=wait(mutex);**

**full=wait(full);**

**empty=signal(empty);**

**printf("\nConsumer consumes item %d",x);**

**x--;**

**mutex=signal(mutex);**

**}**

Q. 94: Parent Child

QUESTION DESCRIPTION  
  
Dani's girlfriend's birthday is near, so he wants to surprise her by making a special cake for her. Dani knows that his girlfriend likes cherries on the cake, so he puts cherries on the top of the cake, but he was not satisfied. If he tries two types of cakes, for baking the cake he needs oven. He has only one oven, so he has to wait until the first cake should get finished. After completion of first cake he should replace that cake with new cake.

TEST CASE 1  
  
INPUT

2

OUTPUT

Processing 1!  
Processing 2!  
Process 1 Enters Critical section  
Process 1 Leaves Critical section  
Process 2 Enters Critical section  
Process 2 Leaves Critical section

TEST CASE 2  
  
INPUT

1

OUTPUT

Processing 1!  
  
Process 1 Enters Critical section  
Process 1 Leaves Critical section

Problem with Question? Report

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int a;**

**cin>>a;**

**if (a>1)**

**{**

**for (int i=1;i<=a;i++)**

**{**

**cout<<"Processing "<<i<<"!"<<endl;**

**}**

**for (int i=1;i<=a;i++)**

**{**

**cout<<"Process "<<i<<" Enters Critical section"<<endl;**

**cout<<"Process "<<i<<" Leaves Critical section"<<endl;**

**}**

**}**

**else**

**{**

**cout<<"Processing 1!"<<endl;**

**cout<<"\n";**

**cout<<"Process 1 Enters Critical section"<<endl;**

**cout<<"Process 1 Leaves Critical section"<<endl;**

**}**

**return 0;**

**}**

Q. 95: Parent Child - 01

QUESTION DESCRIPTION  
  
Raj's girlfriend's birthday is near, so he wants to surprise her by making a special cake for her. He knows that his girlfriend likes cherries on the cake, so he puts cherries on the top of the cake, but he was not satisfied. If he tries two types of cakes, for baking the cake he needs oven. He has only one oven, so he has to wait until the first cake should get finished. After completion of first cake he should replace that cake with new cake.

TEST CASE 1  
  
INPUT

1

OUTPUT

Processing 1!  
  
Process 1 Enters Critical section  
Process 1 Leaves Critical section

TEST CASE 2  
  
INPUT

2

OUTPUT

Processing 1!  
Processing 2!  
Process 1 Enters Critical section  
Process 1 Leaves Critical section  
Process 2 Enters Critical section  
Process 2 Leaves Critical section

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int a;**

**cin>>a;**

**if (a>1)**

**{**

**for (int i=1;i<=a;i++)**

**{**

**cout<<"Processing "<<i<<"!"<<endl;**

**}**

**for (int i=1;i<=a;i++)**

**{**

**cout<<"Process "<<i<<" Enters Critical section"<<endl;**

**cout<<"Process "<<i<<" Leaves Critical section"<<endl;**

**}**

**}**

**else**

**{**

**cout<<"Processing 1!"<<endl;**

**cout<<"\n";**

**cout<<"Process 1 Enters Critical section"<<endl;**

**cout<<"Process 1 Leaves Critical section"<<endl;**

**}**

**return 0;**

**}**

Q. 96: Dining Philoshopers Problem - 02

QUESTION DESCRIPTION  
  
There are 5 friends A,B,C,D,E. They sat at a round table for dinner. To complete dinner each must need 2 Forks . But there are only 5 Forks available on table. They take in such a manner that, first take left Fork and next right Fork. But problem is they try to take at same time. Since they are trying at same time, Fork 1, 2, 3, 4, 5 taken by Friends A,B,C,D,E respectively. And each one tries to get right side Fork. But no one found available Fork. And also that each one thinks that someone will release the Fork and then I can eat. To avoid this situation, Last friend E first try to take right side fork and then left side fork. i.e person E tries to take 4th Fork instead of 5th one. Since 4th Fork already taken by the person D, he gets nothing. But he left 5th Fork. Now the person A will take this 5th Fork and complete dinner and make 1st and 5th available for remaining people. Next the person B takes 1st fork and completes and releases 1st and 2nd. This continuous until all finishes dinner.

TEST CASE 1  
  
INPUT

2

OUTPUT

Fork 1 taken by Philosopher 1  
Fork 2 taken by Philosopher 2  
Fork 3 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Fork 4 taken by Philosopher 1  
Philosopher 2 is waiting for Fork 1  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Philosopher 1 completed his dinner  
Philosopher 1 released fork 1 and fork 4  
Fork 1 taken by Philosopher 2  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 1  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 2 released fork 2 and fork 1  
Fork 2 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 2  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 3 released fork 3 and fork 2  
Fork 3 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Fork 4 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 4 completed his dinner  
Philosopher 4 released fork 4 and fork 3  
  
Till now num of philosophers completed dinner are 4

TEST CASE 2  
  
INPUT

1

OUTPUT

Fork 1 taken by Philosopher 1  
Fork 2 taken by Philosopher 2  
Fork 3 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Fork 4 taken by Philosopher 1  
Philosopher 2 is waiting for Fork 1  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Philosopher 1 completed his dinner  
Philosopher 1 released fork 1 and fork 4  
Fork 1 taken by Philosopher 2  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 1  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 2 released fork 2 and fork 1  
Fork 2 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 2  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 3 released fork 3 and fork 2  
Fork 3 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Fork 4 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 4 completed his dinner  
Philosopher 4 released fork 4 and fork 3  
  
Till now num of philosophers completed dinner are 4

**PROGRAM:**

**#include<stdio.h>**

**#define n 4**

**int compltedPhilo = 0,i;**

**struct fork{**

**int taken;**

**}ForkAvil[n];**

**struct philosp{**

**int left;**

**int right;**

**}Philostatus[n];**

**void goForDinner(int philID){**

**if(Philostatus[philID].left==10 && Philostatus[philID].right==10)**

**printf("Philosopher %d completed his dinner\n",philID+1);**

**else if(Philostatus[philID].left==1 && Philostatus[philID].right==1){**

**printf("Philosopher %d completed his dinner\n",philID+1);**

**Philostatus[philID].left = Philostatus[philID].right = 10;**

**int otherFork = philID-1;**

**if(otherFork== -1)**

**otherFork=(n-1);**

**ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;**

**printf("Philosopher %d released fork %d and fork %d\n",philID+1,philID+1,otherFork+1);**

**compltedPhilo++;**

**}**

**else if(Philostatus[philID].left==1 && Philostatus[philID].right==0){**

**if(philID==(n-1)){**

**if(ForkAvil[philID].taken==0){**

**ForkAvil[philID].taken = Philostatus[philID].right = 1;**

**printf("Fork %d taken by philosopher %d\n",philID+1,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for fork %d\n",philID+1,philID+1);**

**}**

**}else{**

**int dupphilID = philID;**

**philID-=1;**

**if(philID== -1)**

**philID=(n-1);**

**if(ForkAvil[philID].taken == 0){**

**ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;**

**printf("Fork %d taken by Philosopher %d\n",philID+1,dupphilID+1);**

**}else{**

**printf("Philosopher %d is waiting for Fork %d\n",dupphilID+1,philID+1);**

**}**

**}**

**}**

**else if(Philostatus[philID].left==0){**

**if(philID==(n-1)){**

**if(ForkAvil[philID-1].taken==0){**

**ForkAvil[philID-1].taken = Philostatus[philID].left = 1;**

**printf("Fork %d taken by philosopher %d\n",philID,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for fork %d\n",philID+1,philID);**

**}**

**}else{**

**if(ForkAvil[philID].taken == 0){**

**ForkAvil[philID].taken = Philostatus[philID].left = 1;**

**printf("Fork %d taken by Philosopher %d\n",philID+1,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for Fork %d\n",philID+1,philID+1);**

**}**

**}**

**}else{}**

**}**

**int main(){**

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

**ForkAvil[i].taken=Philostatus[i].left=Philostatus[i].right=0;**

**while(compltedPhilo<n){**

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

**goForDinner(i);**

**printf("\nTill now num of philosophers completed dinner are %d\n\n",compltedPhilo);**

**}**

**return 0;**

**}**

Q. 97: Parent Child - 02

QUESTION DESCRIPTION  
  
Dani's girlfriend's birthday is near, so he wants to surprise her by making a special cake for her. Dani knows that his girlfriend likes cherries on the cake, so he puts cherries on the top of the cake, but he was not satisfied. If he tries two types of cakes, for baking the cake he needs oven. He has only one oven, so he has to wait until the first cake should get finished. After completion of first cake he should replace that cake with new cake.

TEST CASE 1  
  
INPUT

2

OUTPUT

Processing 1!  
Processing 2!  
Process 1 Enters Critical section  
Process 1 Leaves Critical section  
Process 2 Enters Critical section  
Process 2 Leaves Critical section

TEST CASE 2  
  
INPUT

1

OUTPUT

Processing 1!  
  
Process 1 Enters Critical section  
Process 1 Leaves Critical section

Problem with Question? Report

**PROGRAM:**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int a;**

**cin>>a;**

**if (a>1)**

**{**

**for (int i=1;i<=a;i++)**

**{**

**cout<<"Processing "<<i<<"!"<<endl;**

**}**

**for (int i=1;i<=a;i++)**

**{**

**cout<<"Process "<<i<<" Enters Critical section"<<endl;**

**cout<<"Process "<<i<<" Leaves Critical section"<<endl;**

**}**

**}**

**else**

**{**

**cout<<"Processing 1!"<<endl;**

**cout<<"\n";**

**cout<<"Process 1 Enters Critical section"<<endl;**

**cout<<"Process 1 Leaves Critical section"<<endl;**

**}**

**return 0;**

**}**

Q. 98: Producer Consumer Problem - 02

QUESTION DESCRIPTION  
  
Alia runs a cheese producing company with minimum storage capacity. Daily She produces some cheese and stores it in her shop. If she produces over than the storage capacity then she will not able to store the product. Once she store on the storage are the consumer can buy the cheese. The product count will reduce whenever the consumer consumes the cheese. If there is no cheese in storage then the consumer cannot buy any product.

TEST CASE 1  
  
INPUT

1  
2  
2  
3

OUTPUT

Producer produces the item 1  
Consumer consumes item 1  
Buffer is empty!!

TEST CASE 2  
  
INPUT

2  
3

OUTPUT

Buffer is empty!!

**PROGRAM:**

**#include<stdio.h>**

**#include<stdlib.h>**

**int mutex=1,full=0,empty=3,x=0;**

**int main()**

**{**

**int n;**

**void producer();**

**void consumer();**

**int wait(int);**

**int signal(int);**

**while(1)**

**{**

**scanf("%d",&n);**

**switch(n)**

**{**

**case 1: if((mutex==1)&&(empty!=0))**

**producer();**

**else**

**printf("\nBuffer is full!!");**

**break;**

**case 2: if((mutex==1)&&(full!=0))**

**consumer();**

**else**

**printf("\nBuffer is empty!!");**

**break;**

**case 3:**

**exit(0);**

**break;**

**}**

**}**

**return 0;**

**}**

**int wait(int s)**

**{**

**return (--s);**

**}**

**int signal(int s)**

**{**

**return(++s);**

**}**

**void producer()**

**{**

**mutex=wait(mutex);**

**full=signal(full);**

**empty=wait(empty);**

**x++;**

**printf("\nProducer produces the item %d",x);**

**mutex=signal(mutex);**

**}**

**void consumer()**

**{**

**mutex=wait(mutex);**

**full=wait(full);**

**empty=signal(empty);**

**printf("\nConsumer consumes item %d",x);**

**x--;**

**mutex=signal(mutex);**

**}**

Q. 99: Dining Philoshopers

QUESTION DESCRIPTION  
  
The dining philosophers problem is a classic example in computer science often used to illustrate synchronization issues and solutions in concurrent algorithm design. It illustrates the challenges of avoiding a system state where progress is not possible, a deadlock. In Operating System, this concept used in process synchronization. Same problem but instead of Philosophers processes are there and instead of Forks Resources are there. There are 5 friends A,B,C,D,E. They sat at a round table for dinner. To complete dinner each must need 2 Forks . But there are only 5 Forks available on table. They take in such a manner that, first take left Fork and next right Fork. But problem is they try to take at same time. Since they are trying at same time, Fork 1, 2, 3, 4, 5 taken by Friends A,B,C,D,E respectively. And each one tries to get right side Fork. But no one found available Fork. And also that each one thinks that someone will release the Fork and then I can eat. To avoid this situation, Last friend E first try to take right side fork and then left side fork. i.e person E tries to take 4th Fork instead of 5th one. Since 4th Fork already taken by the person D, he gets nothing. But he left 5th Fork. Now the person A will take this 5th Fork and complete dinner and make 1st and 5th available for remaining people. Next the person B takes 1st fork and completes and releases 1st and 2nd. This continuous until all finishes dinner.

TEST CASE 1  
  
INPUT

2

OUTPUT

Fork 1 taken by Philosopher 1  
Fork 2 taken by Philosopher 2  
Fork 3 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Fork 4 taken by Philosopher 1  
Philosopher 2 is waiting for Fork 1  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Philosopher 1 completed his dinner  
Philosopher 1 released fork 1 and fork 4  
Fork 1 taken by Philosopher 2  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 1  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 2 released fork 2 and fork 1  
Fork 2 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 2  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 3 released fork 3 and fork 2  
Fork 3 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Fork 4 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 4 completed his dinner  
Philosopher 4 released fork 4 and fork 3  
  
Till now num of philosophers completed dinner are 4

TEST CASE 2  
  
INPUT

1

OUTPUT

Fork 1 taken by Philosopher 1  
Fork 2 taken by Philosopher 2  
Fork 3 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Fork 4 taken by Philosopher 1  
Philosopher 2 is waiting for Fork 1  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 0  
  
Philosopher 1 completed his dinner  
Philosopher 1 released fork 1 and fork 4  
Fork 1 taken by Philosopher 2  
Philosopher 3 is waiting for Fork 2  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 1  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 2 released fork 2 and fork 1  
Fork 2 taken by Philosopher 3  
Philosopher 4 is waiting for fork 3  
  
Till now num of philosophers completed dinner are 2  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 3 released fork 3 and fork 2  
Fork 3 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Fork 4 taken by philosopher 4  
  
Till now num of philosophers completed dinner are 3  
  
Philosopher 1 completed his dinner  
Philosopher 2 completed his dinner  
Philosopher 3 completed his dinner  
Philosopher 4 completed his dinner  
Philosopher 4 released fork 4 and fork 3  
  
Till now num of philosophers completed dinner are 4

**PROGRAM:**

**#include<stdio.h>**

**#define n 4**

**int compltedPhilo = 0,i;**

**struct fork{**

**int taken;**

**}ForkAvil[n];**

**struct philosp{**

**int left;**

**int right;**

**}Philostatus[n];**

**void goForDinner(int philID){**

**if(Philostatus[philID].left==10 && Philostatus[philID].right==10)**

**printf("Philosopher %d completed his dinner\n",philID+1);**

**else if(Philostatus[philID].left==1 && Philostatus[philID].right==1){**

**printf("Philosopher %d completed his dinner\n",philID+1);**

**Philostatus[philID].left = Philostatus[philID].right = 10;**

**int otherFork = philID-1;**

**if(otherFork== -1)**

**otherFork=(n-1);**

**ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;**

**printf("Philosopher %d released fork %d and fork %d\n",philID+1,philID+1,otherFork+1);**

**compltedPhilo++;**

**}**

**else if(Philostatus[philID].left==1 && Philostatus[philID].right==0){**

**if(philID==(n-1)){**

**if(ForkAvil[philID].taken==0){**

**ForkAvil[philID].taken = Philostatus[philID].right = 1;**

**printf("Fork %d taken by philosopher %d\n",philID+1,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for fork %d\n",philID+1,philID+1);**

**}**

**}else{**

**int dupphilID = philID;**

**philID-=1;**

**if(philID== -1)**

**philID=(n-1);**

**if(ForkAvil[philID].taken == 0){**

**ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;**

**printf("Fork %d taken by Philosopher %d\n",philID+1,dupphilID+1);**

**}else{**

**printf("Philosopher %d is waiting for Fork %d\n",dupphilID+1,philID+1);**

**}**

**}**

**}**

**else if(Philostatus[philID].left==0){**

**if(philID==(n-1)){**

**if(ForkAvil[philID-1].taken==0){**

**ForkAvil[philID-1].taken = Philostatus[philID].left = 1;**

**printf("Fork %d taken by philosopher %d\n",philID,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for fork %d\n",philID+1,philID);**

**}**

**}else{**

**if(ForkAvil[philID].taken == 0){**

**ForkAvil[philID].taken = Philostatus[philID].left = 1;**

**printf("Fork %d taken by Philosopher %d\n",philID+1,philID+1);**

**}else{**

**printf("Philosopher %d is waiting for Fork %d\n",philID+1,philID+1);**

**}**

**}**

**}else{}**

**}**

**int main(){**

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

**ForkAvil[i].taken=Philostatus[i].left=Philostatus[i].right=0;**

**while(compltedPhilo<n){**

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

**goForDinner(i);**

**printf("\nTill now num of philosophers completed dinner are %d\n\n",compltedPhilo);**

**}**

**return 0;**

**}**

Q. 100: Producer Consumer Problem - 01

QUESTION DESCRIPTION  
  
Geetha runs a bread manufacturing company with limited storage capacity. Daily She manufactures small amount of bread and stores it in her shop. If she manufactures over than the storage capacity then she will not able to store the product. Once she store on the storage are the consumer can buy the bread. The product count will reduce whenever the consumer consumes the bread. If there is no bread in storage then the consumer cannot buy any product.

TEST CASE 1  
  
INPUT

2  
3

OUTPUT

Buffer is empty!!

TEST CASE 2  
  
INPUT

1  
2  
3  
4

OUTPUT

Producer produces the item 1  
Consumer consumes item 1

**PROGRAM:**

**#include<stdio.h>**

**#include<stdlib.h>**

**int mutex=1,full=0,empty=3,x=0;**

**int main()**

**{**

**int n;**

**void producer();**

**void consumer();**

**int wait(int);**

**int signal(int);**

**while(1)**

**{**

**scanf("%d",&n);**

**switch(n)**

**{**

**case 1: if((mutex==1)&&(empty!=0))**

**producer();**

**else**

**printf("\nBuffer is full!!");**

**break;**

**case 2: if((mutex==1)&&(full!=0))**

**consumer();**

**else**

**printf("\nBuffer is empty!!");**

**break;**

**case 3:**

**exit(0);**

**break;**

**}**

**}**

**return 0;**

**}**

**int wait(int s)**

**{**

**return (--s);**

**}**

**int signal(int s)**

**{**

**return(++s);**

**}**

**void producer()**

**{**

**mutex=wait(mutex);**

**full=signal(full);**

**empty=wait(empty);**

**x++;**

**printf("\nProducer produces the item %d",x);**

**mutex=signal(mutex);**

**}**

**void consumer()**

**{**

**mutex=wait(mutex);**

**full=wait(full);**

**empty=signal(empty);**

**printf("\nConsumer consumes item %d",x);**

**x--;**

**mutex=signal(mutex);**

**}**

**END**