**Student Name: Naveen Singh**

**Student ID : 11702152**

**Email Address: naveensingh8783@gmail.com**

**GitHub Link:**

**CODE:**

#include<stdio.h>

void resoure\_allocation(int e,int f,int g,int h); //method to allocate resources

void max\_resource\_need(int a,int b,int c,int d); //method for maximum resources required

int allocation[10][10], max[10][10];

int main()

{

int process=5; //Total number of process

int resource=3; //Total number of resources

int need[10][10], Available\_resource[10],i,j,k,initial[10],check=0,safe\_sequence[10],z=0,count;

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

{

initial[i]=0; //none of the process is satisfied initially

}

// Allocating the Resources to processes

resoure\_allocation(0,0,1,0);

resoure\_allocation(1,2,0,0);

resoure\_allocation(2,3,0,2);

resoure\_allocation(3,2,1,1);

resoure\_allocation(4,0,0,2);

// Maximum Resource Required to processes

max\_resource\_need(0,7,5,3);

max\_resource\_need(1,3,2,2);

max\_resource\_need(2,9,0,2);

max\_resource\_need(3,2,2,2);

max\_resource\_need(4,4,3,3);

// Available Resource

Available\_resource[0]=3;

Available\_resource[1]=3;

Available\_resource[2]=2;

//Display allocated resource

printf("Resource allocate\n");

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

{

printf("P[%d] ",i);

for(j=0;j<resource;j++)

{

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

}

printf("\n");

}

// Display maximum Resource Requried

printf("\nMaximum Resource Required\n");

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

{

printf("P[%d] ",i);

for(j=0;j<resource;j++)

{

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

}

printf("\n");

}

//need of each process

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

{

for(j=0;j<3;j++)

{

need[i][j]=max[i][j]-allocation[i][j];

}

}

printf("\n");

//Display needs of process

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

{

printf("Need of P[%d] is: ",i);

for(j=0;j<resource;j++)

{

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

}

printf("\n");

}

printf("\n");

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

{

for(j=0;j<process;j++) //for process

{

count=0;

for(k=0;k<resource;k++) //for resource

{

if(initial[j]==0 && need[j][k]<=Available\_resource[k]) //need[j][k] means need of jth process's kth resource

{

Available\_resource[k]=Available\_resource[k]+allocation[j][k];

count++;

if(count==resource) //if true means need for all the resources can be satisfied

{

initial[j]=1; // process has been allocated resources hence initial array value

printf("Process %d has been allocated all required resources\n",j);

safe\_sequence[z]=j; // array sequence stores the safe sequence. jth process is added to safe sequence

z++;

}

continue; //if need of one resource is satisfied then we check for next resource

}

break; // if need for any resource of a process can not be satisfied, break the loop and check the

//need of next process

}

}

}

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

{

if(initial[i]==0)

{

printf("\nsystem is in unsafe state\n");

check=1;

break;

}

}

if(check==0)

{

printf("\nSystem is in safe state\n");

printf("\nSafe sequence is \n");

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

{

printf("P[%d],",safe\_sequence[i]);

}

}

printf("\n");

return 0;

}

void resoure\_allocation(int e,int f,int g,int h)

{

allocation[e][0]=f;

allocation[e][1]=g;

allocation[e][2]=h;

}

void max\_resource\_need(int a,int b,int c,int d)

{

max[a][0]=b;

max[a][1]=c;

max[a][2]=d;

}

**Description:**

In this problem 5 process are given to us with allocated resources and maximum resources each process required to complete its task. We are provided with 3 resources with instances that are 10 in resource A, 5 in resource B and 7 in resource C. Among the given instances some of the instances is allocated to different resources. Since all process required more instances of different resource type to complete its task according to given table data.

Given data:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Available | | | Processes | Allocation | | | Max | | |
| A | B | C | A | B | C | A | B | C |
| 3 | 3 | 2 | P0 | 0 | 1 | 0 | 7 | 5 | 3 |
|  | | | P1 | 2 | 0 | 0 | 3 | 2 | 2 |
| P2 | 3 | 0 | 2 | 9 | 0 | 2 |
| P3 | 2 | 1 | 1 | 2 | 2 | 2 |
| P4 | 0 | 0 | 2 | 4 | 3 | 3 |

Here P0 is allocated with 1 instance of resource B but it require 7 instances of resource A, 5 instance of resource B and 3 instance of resource C hence to complete its task P0 still require 7 instance of resource A 4 instance of resource B and 3 instance of resource C. In the same way P1,P2....P5 need more instances of different resources. According to question we are required to build a system which can tell if this scenario is in safe state or in unsafe state means can lead to deadlock or not.

**Implementation , Algorithm used and complexity of different function loops used:**

We can implement this scenario by using Banker’s Algorithm.

Steps to implement

Allocate all the required information according to given data in table.

Process=5,resource=3.call these 2 function to allocate resouces and maximum resource required to process with given parameter. These function having complexity O(1).

void resoure\_allocation(int e,int f,int g,int h)

{

allocation[e][0]=f;

allocation[e][1]=g;

allocation[e][2]=h;

}

void max\_resource\_need(int a,int b,int c,int d)

{

max[a][0]=b;

max[a][1]=c;

max[a][2]=d;

}

\*This loop ensure that no process initially have any resource’s instances with complexity O(n)

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

{

initial[i]=0; //none of the process is satisfied initially

}

\*This loop will calculate the need of process with complexity O(n).

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

{

for(j=0;j<3;j++)

{

need[i][j]=max[i][j]-allocation[i][j];

}

}

\*These loops Display the Information with complexity O(n)

Used 3 time to display information in max[i][j],allocation[i][j],need[i][j] array individually

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

{

printf("P[%d] ",i);

for(j=0;j<resource;j++)

{

printf("%d ",max[i][j]); //allocation[i][j],//need[i][j]

}

printf("\n");

}

\*This is main loop of this program which allocating the resource by checking some condition and also find the safe sequence in which these process has to be allocated resource’s instance if possible with complexity O(pow(n,3)).

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

{

for(j=0;j<process;j++) //for process

{

count=0;

for(k=0;k<resource;k++) //for resource

{

if(initial[j]==0 && need[j][k]<=Available\_resource[k]) //need[j][k] means need of jth process's kth resource

{

Available\_resource[k]=Available\_resource[k]+allocation[j][k];

count++;

if(count==resource) //if true means need for all the resources can be satisfied

{

initial[j]=1; // process has been allocated resources hence initial array value

printf("Process %d has been allocated all required resources\n",j);

safe\_sequence[z]=j; // array sequence stores the safe sequence. jth process is added to safe sequence

z++;

}

continue; //if need of one resource is satisfied then we check for next resource

}

break; // if need for any resource of a process can not be satisfied, break the loop and check the

//need of next process

}

}

}

\*This loop is checking whether the system is in safe state or not and if it is not in safe state it will display the message with complexity O(n)

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

{

if(initial[i]==0)

{

printf("\nsystem is in unsafe state\n");

check=1;

break;

}

}

\*This loop display the safe state massage and safe sequence with complexity O(n)

if(check==0)

{

printf("\nSystem is in safe state\n");

printf("\nSafe sequence is \n");

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

{

printf("P[%d],",safe\_sequence[i]);

}

}

**Overall complexity of the programme is O(pow(n,3)).**

**OUTPUT:**

**Resource allocate**

**P[0] 0 1 0**

**P[1] 2 0 0**

**P[2] 3 0 2**

**P[3] 2 1 1**

**P[4] 0 0 2**

**Maximum Resource Required**

**P[0] 7 5 3**

**P[1] 3 2 2**

**P[2] 9 0 2**

**P[3] 2 2 2**

**P[4] 4 3 3**

**Need of P[0] is: 7 4 3**

**Need of P[1] is: 1 2 2**

**Need of P[2] is: 6 0 0**

**Need of P[3] is: 0 1 1**

**Need of P[4] is: 4 3 1**

**Process 1 has been allocated all required resources**

**Process 3 has been allocated all required resources**

**Process 4 has been allocated all required resources**

**Process 0 has been allocated all required resources**

**Process 2 has been allocated all required resources**

**System is in safe state**

**Safe sequence is**

**P[1],P[3],P[4],P[0],P[2],**

**--------------------------------**

**Process exited after 0.1783 seconds with return value 0**

**Press any key to continue . . .**