**Assignment 8**

**Aim –**

Simulate of Banker’s algorithm using c language.

Code :

#include<stdio.h>

void main()

{ //Here we need 3 arrays namely Allocation, Max and Available

// arrays alloc and max are 2D arrays. array 'available' is a 1D array.

int n, m, i, j, k, alloc[20][20], max[20][20], available[20];

int f[20],ans[20], ind=0, need[20][20]; //We need the Need matrix.

//Reading the number of Processes from the input.

printf("Enter number of processes: ");

scanf("%d",&n);

//Reading the number of Resources from the input.

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

scanf("%d",&m);

//Reading the Allocation Values to the Matrix 'alloc[][]'

printf("Enter the Values of Allocation Matrix: \n");

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

{

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

{

printf("Enter value at position (%d%d) :",i+1,j+1);

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

}

}

//Reading the Max values to the matrix 'max[][]'

printf("Enter the Values of Max Matrix: \n");

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

{

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

{

printf("Enter value at position (%d%d) :",i+1,j+1);

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

}

}

//Reading the values of array available[]

printf("Enter the values in available: \n");

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

{

printf("Enter value at position (%d) :",j+1);

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

}

//We are using an array f to represent the finished status of each process.

//Initially setting all processes as not finished. ie. Setting f[i]=0 for each process i.

for(k=0;k<n;k++)

{

f[k]=0;

}

//Calculating values of the NEED MATRIX using its equation, for all processes

//Equation is need[i][j] = max[i][j] - allocation[i][j]

for(i=0;i<n;i++) //For each process

{

for(j=0;j<m;j++) //For each resource

{

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

}

}

//Finding safe sequence

int y=0;

// ans[] array will be used to store the SAFE SEQUENCE in the end.

for(k=0;k<n;k++)

{

for(i=0;i<n;i++) //For each process

{

if(f[i]==0)

{

int flag = 0; //setting flag as 0 or false.

for(j=0;j<m;j++) //For each Resource

{

if(need[i][j] > available[j]) //If Need greater than Available, then

{

flag=1; //Setting flag as true or 1.

//flag=1 means the Need is greater than what is Available for that particular resource.

break; //Breaking out of this loop if need > available

}

}

if(flag==0)

{

ans[ind++] = i;

for(y=0;y<m;y++) //For each Resource

{

available[y] = available[y] + alloc[i][y]; //Setting availability to current availability + allocation

}

f[i]=1; //Declaring the current process as FINISHED.

}

}

}

}

//Displaying the SAFE SEQUENCE.

printf("The SAFE SEQUENCE is: \n");

for(i=0;i<n-1;i++) //Here loop ends at n-1 because we don't want to printf the arrowmark(->) at the end.printf(" P%d", answer[n-1)

{

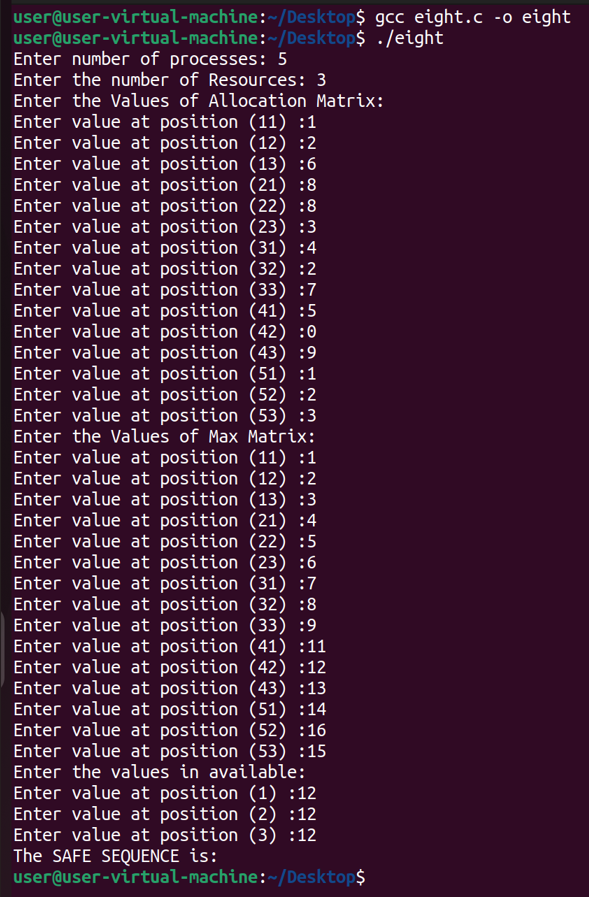
printf(" P%d ->", ans[i]);

}

printf(" P%d", ans[n-1]); //Printing the final state in safe sequence without printing the arrowmark.

}

OUTPUT :



Conclusion : Hence we have successfully studied and executed the Bankers algorithm.

**THANK YOU!!!!!!!!!!**