**CODE >>**

#include<stdio.h>

#define NUM\_PROC 10

#define NUM\_RES 10

void input1D(int arr[], int n)

{

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

{

printf("\nEnter Total Resource %d >> ",j+1);

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

}

}

void input2D(int arr[][NUM\_RES], int n, int m)

{

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

{

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

{

printf("\nEnter resource %d for Process %d >> ",j+1,i+1);

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

}

}

}

void display1D(int arr[], int m)

{

printf("\n");

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

{

printf("%d\t", arr[j]);

}

printf("\n");

}

void display2D(int arr[][NUM\_RES], int m, int n)

{

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

{

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

{

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

}

printf("\n");

}

printf("\n");

}

void calculateCurrAvail(int alloct[][NUM\_RES], int total[], int curAvail[], int m, int n)

{

int temp[n];

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

{

temp[i]=0;

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

{

temp[i] += alloct[j][i];

}

curAvail[i] = total[i] - temp[i];

}

}

void calculateNeed(int maxm[][NUM\_RES], int alloct[][NUM\_RES], int need[][NUM\_RES], int m, int n)

{

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

{

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

{

need[i][j] = maxm[i][j] - alloct[i][j];

}

}

}

void requestProcess(int request[], int curAvl[], int alloc[][NUM\_RES], int need[][NUM\_RES], int n)

{

int ch,process,flag=0;

printf("\nDo you want to add request to process >> ");

scanf("%d",&ch);

if(ch==1)

{

printf("\nFor which process you want to give request >> ");

scanf("%d",&process);

printf("\nEnter the request resources for process %d >> ",process);

input1D(request,n);

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

{

if(request[i] <= need[process][i] && request[i] <= curAvl[i])

{

flag=0;

}

else

{

flag=1;

break;

}

}

if(flag!=1)

{

printf("\nThe request is accepted!! ");

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

{

curAvl[i] -= request[i];

need[process][i] -= request[i];

alloc[process][i] += request[i];

}

}

else

{

printf("\nThe request cannot be accepted!!! ");

}

}

else

{

return;

}

}

void calculateFinalAvailable(int alloc[][NUM\_RES], int need[][NUM\_RES], int curAvl[], int finalAvl[][NUM\_RES], int m, int n)

{

int count = m;

int arr[NUM\_PROC];

int done[NUM\_PROC] = {0};

int i = 0;

while (count > 0)

{

int flag = 0;

int currProc = -1;

for (int p = 0; p < m; p++)

{

if (done[p] == 0)

{

int canAllocate = 1;

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

{

if (need[p][j] > curAvl[j])

{

canAllocate = 0;

break;

}

}

if (canAllocate)

{

currProc = p;

break;

}

}

}

if (currProc == -1)

{

printf("No safe sequence found.\n");

return;

}

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

{

curAvl[j] += alloc[currProc][j];

finalAvl[currProc][j] = curAvl[j];

}

done[currProc] = 1;

arr[i++] = currProc;

count--;

}

printf("\n");

printf("Safe sequence: P%d", arr[0]);

for (int i = 1; i < m; i++)

{

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

}

printf("\n");

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

{

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

{

printf("%d\t", finalAvl[arr[i]][j]);

}

printf("\n");

}

printf("\n");

}

int main()

{

int numProc=5, numRes=4;

printf("\nEnter the Number of Processes >> ");

scanf("%d",&numProc);

printf("\nEnter the Number of Resources >> ");

scanf("%d",&numRes);

int alloct[NUM\_PROC][NUM\_RES] = {{0, 0, 1, 2}, {2, 0, 0, 0}, {0, 0, 3, 4}, {2, 3, 5, 4}, {0, 3, 3, 2}};

int maxm[NUM\_PROC][NUM\_RES] = {{0, 0, 1, 2}, {2, 7, 5, 0}, {6, 6, 5, 6}, {4, 3, 5, 6}, {0, 6, 5, 2}};

int totalRes[NUM\_RES] = {6, 7, 12, 12};

int avaInitial[NUM\_RES], need[NUM\_PROC][NUM\_RES], avaFinal[NUM\_PROC][NUM\_RES];

int request[NUM\_RES] = {1,0,2};

printf("\nEnter values for Matrix Allocation >> ");

input2D(alloct,numProc,numRes);

printf("\nEnter values for Matrix Max-Claim >> ");

input2D(maxm,numProc,numRes);

printf("\nEnter values for Matrix Total Resource >> ");

input1D(totalRes,numRes);

printf("\nCalculating Currently Available Matrix");

calculateCurrAvail(alloct,totalRes,avaInitial,numProc,numRes);

printf("\nCalculating Need Matrix");

calculateNeed(maxm,alloct,need,numProc,numRes);

requestProcess(request,avaInitial,alloct,need,numRes);

printf("\nDisplaying Max-Claim Matrix >> \n");

display2D(maxm,numProc,numRes);

printf("\nDisplaying Allocated Matrix >> \n");

display2D(alloct,numProc,numRes);

printf("\nDisplaying Need Matrix >> \n");

display2D(need,numProc,numRes);

printf("\nDisplaying Currently Avilable Matrix >> \n");

display1D(avaInitial,numRes);

printf("\nDisplaying Total Resource Matrix >> \n");

display1D(totalRes,numRes);

printf("\nCalculating Final Available Matrix");

calculateFinalAvailable(alloct,need,avaInitial,avaFinal,numProc,numRes);

return 0;

}

**OUTPUT >>**

student@student:~/Downloads$ gcc bankers.c

student@student:~/Downloads$ ./a.out

Enter the Number of Processes >> 5

Enter the Number of Resources >> 3

Enter values for Matrix Allocation >>

Enter resource 1 for Process 1 >> 0

Enter resource 2 for Process 1 >> 1

Enter resource 3 for Process 1 >> 0

Enter resource 1 for Process 2 >> 2

Enter resource 2 for Process 2 >> 0

Enter resource 3 for Process 2 >> 0

Enter resource 1 for Process 3 >> 3

Enter resource 2 for Process 3 >> 0

Enter resource 3 for Process 3 >> 2

Enter resource 1 for Process 4 >> 2

Enter resource 2 for Process 4 >> 1

Enter resource 3 for Process 4 >> 1

Enter resource 1 for Process 5 >> 0

Enter resource 2 for Process 5 >> 0

Enter resource 3 for Process 5 >> 2

Enter values for Matrix Max-Claim >>

Enter resource 1 for Process 1 >> 7

Enter resource 2 for Process 1 >> 5

Enter resource 3 for Process 1 >> 3

Enter resource 1 for Process 2 >> 3

Enter resource 2 for Process 2 >> 2

Enter resource 3 for Process 2 >> 2

Enter resource 1 for Process 3 >> 9

Enter resource 2 for Process 3 >> 0

Enter resource 3 for Process 3 >> 2

Enter resource 1 for Process 4 >> 2

Enter resource 2 for Process 4 >> 2

Enter resource 3 for Process 4 >> 2

Enter resource 1 for Process 5 >> 4

Enter resource 2 for Process 5 >> 3

Enter resource 3 for Process 5 >> 3

Enter values for Matrix Total Resource >>

Enter Total Resource 1 >> 10

Enter Total Resource 2 >> 5

Enter Total Resource 3 >> 7

Calculating Currently Available Matrix

Calculating Need Matrix

Do you want to add request to process >> 1

For which process you want to give request >> 1

Enter the request resources for process 1 >>

Enter Total Resource 1 >> 1

Enter Total Resource 2 >> 0

Enter Total Resource 3 >> 2

The request is accepted!!

Displaying Max-Claim Matrix >>

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Displaying Allocated Matrix >>

0 1 0

3 0 2

3 0 2

2 1 1

0 0 2

Displaying Need Matrix >>

7 4 3

0 2 0

6 0 0

0 1 1

4 3 1

Displaying Currently Avilable Matrix >>

2 3 0

Displaying Total Resource Matrix >>

10 5 7

Calculating Final Available Matrix

Safe sequence: P1 -> P3 -> P0 -> P2 -> P4

5 3 2

7 4 3

7 5 3

10 5 5

10 5 7

student@student:~/Downloads$