Assignment 5: Deadlock Avoidance & detection Algorithm

# Name – Akshat Chandrapatle Div- CS-A Roll no.- 45 PRN- 12111449

Bankers algorithm

// Banker's Algorithm

#include <stdio.h>

int main()

{

// P0, P1, P2, P3, P4 are the Process names here

int n, m, i, j, k;

n = 0; // Number of processes

m = 0; // Number of resources

printf("Enter number of processes: ");

scanf("%d",&n);

printf("\nEnter number of resources ");

scanf("%d",&m);

int max[n][m];

int alloc[n][m];

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

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

alloc[i][j]=0;

}

}

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

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

max[i][j]=0;

}

}

int avail[m];

printf("Enter available resources: \n");

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

printf("R%d",i+1);

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

}

printf("Enter Allocation matrix: \n");

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

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

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

}

}

printf("Enter Max matrix: \n");

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

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

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

}

}

int f[n], ans[n], ind = 0;

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

f[k] = 0;

}

int need[n][m];

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

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

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

}

int y = 0;

for (k = 0; k < 5; k++) {

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

if (f[i] == 0) {

int flag = 0;

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

if (need[i][j] > avail[j]){

flag = 1;

break;

}

}

if (flag == 0) {

ans[ind++] = i;

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

avail[y] += alloc[i][y];

f[i] = 1;

}

}

}

}

int flag = 1;

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

{

if(f[i]==0)

{

flag=0;

printf("The following system is not safe");

break;

}

}

if(flag==1)

{

printf("Following is the SAFE Sequence\n");

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

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

printf(" P%d", ans[n - 1]);

}

return (0);

}

**OUTPUT:**