**LAB-6**

**Deadlock problem**

1. Write a C program to simulate Bankers algorithm for the purpose of

deadlock avoidance.

#include <stdio.h>

int main() {

int n, m, i, j, k;

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

scanf("%d", &n);

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

scanf("%d", &m);

int allocation[n][m];

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

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

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

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

}

)

int max[n][m];

printf("Enter the MAX Matrix:\n");

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

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

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

}

}

int available[m];

printf("Enter the Available Resources:\n");

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

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

}

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] - allocation[i][j];

}

}

int y = 0;

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

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

if (f[i] == 0) {

int flag = 0;

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

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

flag = 1;

break;

}

}

if (flag == 0) {

ans[ind++] = i;

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

available[y] += allocation[i][y];

}

f[i] = 1;

}

}

}

}

int flag = 1;

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

if (f[i] == 0) {

flag = 0;

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

break;

}

}

if (flag == 1) {

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

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

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

}

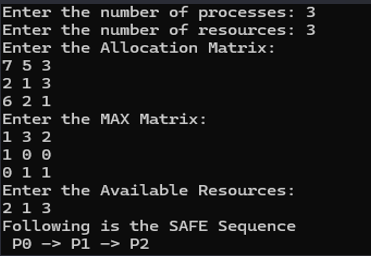
printf(" P%d\n", ans[n - 1]);

}

return 0;

}

OUTPUT:



2. Write a C program to simulate deadlock detection.

#include <stdio.h>  
static int mark[20];  
int i, j, np, nr;  
int main()  
{  
    int alloc[10][10], request[10][10], avail[10], r[10], w[10];  
    printf("\nEnter the no of process: ");  
    scanf("%d", &np);  
    printf("\nEnter the no of resources: ");  
    scanf("%d", &nr);  
    for(i = 0; i < nr; i++)  
    {  
        printf("\nTotal Amount of the Resource R%d: ", i + 1);  
        scanf("%d", &r[i]);  
    }  
    printf("\nEnter the request matrix:");  
    for(i = 0; i < np; i++)  
        for(j = 0; j < nr; j++)  
            scanf("%d", &request[i][j]);  
  printf("\nEnter the allocation matrix:");  
    for(i = 0; i < np; i++)  
        for(j = 0; j < nr; j++)  
            scanf("%d", &alloc[i][j]);  
      for(j = 0; j < nr; j++)  
    {  
        avail[j] = r[j];  
        for(i = 0; i < np; i++)  
        {  
            avail[j] -= alloc[i][j];  
        }  
    }   
    for(i = 0; i < np; i++)  
    {  
        int count = 0;  
        for(j = 0; j < nr; j++)  
        {  
            if(alloc[i][j] == 0)  
                count++;  
            else  
                break;  
        }  
        if(count == nr)  
            mark[i] = 1;  
    }  
  for(j = 0; j < nr; j++)  
        w[j] = avail[j];   
    for(i = 0; i < np; i++)  
    {  
        int canbeprocessed = 0;  
        if(mark[i] != 1)  
        {  
            for(j = 0; j < nr; j++)  
            {  
                if(request[i][j] <= w[j])  
                    canbeprocessed = 1;  
                else  
                {  
                    canbeprocessed = 0;  
                    break;  
                }  
            }  
            if(canbeprocessed)  
            {  
                mark[i] = 1;  
                for(j = 0; j < nr; j++)  
                    w[j] += alloc[i][j];  
            }  
        }  
    }   
    int deadlock = 0;  
    for(i = 0; i < np; i++)  
        if(mark[i] != 1)  
            deadlock = 1;  
    if(deadlock)  
        printf("\n Deadlock detected");  
    else  
        printf("\n No Deadlock possible");  
}

OUTPUT:

