**NAME: DHANRAJ SUBHASH KORE**

**DIV: TY B** **BATCH: B3**

**ROLL NO: 60**

**SUB: SS LAB ASSIGNMENT**

**Bankers Algorithm**

Implement banker's algorithm

**C CODE:**

#include <stdio.h>

#include <stdlib.h>

void print(int x[][10], int n, int m)

{

        int i, j;

        for (i = 0; i < n; i++)

        {

                printf("\n");

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

                {

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

                }

        }

}

int safety(int A[][10], int N[][10], int AV[1][10], int n, int m, int a[])

{

        int i, j, k, x = 0;

        int F[10], W[1][10];

        int pflag = 0, flag = 0;

        for (i = 0; i < n; i++)

                F[i] = 0;

        for (i = 0; i < m; i++)

                W[0][i] = AV[0][i];

        for (k = 0; k < n; k++)

        {

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

                {

                        if (F[i] == 0)

                        {

                                flag = 0;

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

                                {

                                        if (N[i][j] > W[0][j])

                                                flag = 1;

                                }

                                if (flag == 0 && F[i] == 0)

                                {

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

                                                W[0][j] += A[i][j];

                                        F[i] = 1;

                                        pflag++;

                                        a[x++] = i;

                                }

                        }

                }

                if (pflag == n)

                        return 1;

        }

        return 0;

}

void accept(int A[][10], int N[][10], int M[10][10], int W[1][10], int \*n, int \*m)

{

        int i, j;

        printf("\n Enter total no. of processes : ");

        scanf("%d", n);

        printf("\n Enter total no. of resources : ");

        scanf("%d", m);

        for (i = 0; i < \*n; i++)

        {

                printf("\n Process %d\n", i + 1);

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

                {

                        printf(" Allocation for resource %d : ", j + 1);

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

                        printf(" Maximum for resource %d : ", j + 1);

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

                }

        }

        printf("\n Available resources : \n");

        for (i = 0; i < \*m; i++)

        {

                printf(" Resource %d : ", i + 1);

                scanf("%d", &W[0][i]);

        }

        for (i = 0; i < \*n; i++)

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

                        N[i][j] = M[i][j] - A[i][j];

        printf("\n Allocation Matrix : ");

        print(A, \*n, \*m);

        printf("\n\n Maximum Requirement Matrix : ");

        print(M, \*n, \*m);

        printf("\n\n Need Matrix : ");

        print(N, \*n, \*m);

}

int banker(int A[][10], int N[][10], int W[1][10], int n, int m)

{

        int j, i, a[10];

        j = safety(A, N, W, n, m, a);

        if (j != 0)

        {

                printf("\n\n");

                printf("\n A safety sequence has been detected!\n");

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

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

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

                return 1;

        }

        else

        {

                printf("\n Deadlock has occured!\n");

                return 0;

        }

}

int main()

{

        int A[10][10];

        int M[10][10];

        int N[10][10];

        int W[1][10];

        int n, m, pid, ch;

        accept(A, N, M, W, &n, &m);

        banker(A, N, W, n, m);

        return 0;

}

**O/P:**







