**EXP NO :9 DEADLOCK PREVENTION AND AVOIDANCE ALGORIITHM**

**AIM:**

To write a C program to reduce the deadlock using Banker’s algorithm in deadlock prevention.

**ALGORITHM:**

**STEP 1:** Start the program.

**STEP 2:** Let work and finish be the vertex of length m and n respectively.

**STEP 3:** Initialise work is equal to available and finish[1] = false for i=0,1,…n-1.

**STEP 4:**

1. Find an index I such that both
2. Finish[i] =false;
3. Need[i]<= work;
4. If no such I exist, goto step 4.
5. Work = work + allocation

Finish[i] = true

Goto step 2.

1. If finish[i] == true for all I, then the system is in the safe state.

**STEP 5:** Request I am the vertex for process pi.

If request[i] == k,

1. if request<= head, goto step 2 otherwise rise an error condition that the process has exceeded its max chain.
2. If request <= available, goto step 3, otherwise Pi must wait, since the resource is not available.

**STEP 6:** Stop the program.

**CODE:**

#include<stdio.h>

#include<conio.h>

int main()

{

int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10], safeSequence[10];

int p, r, i, j, process, count;

count = 0;

printf("Enter the no of processes : ");

scanf("%d", &p);

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

completed[i] = 0;

printf("\n\nEnter the no of resources : ");

scanf("%d", &r);

printf("\n\nEnter the Max Matrix for each process : ");

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

{

printf("\nFor process %d : ", i + 1);

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

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

}

printf("\n\nEnter the allocation for each process : ");

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

{

printf("\nFor process %d : ",i + 1);

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

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

}

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

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

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

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

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

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

do

{

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

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

{

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

printf("%d ", Max[i][j]);

printf("\t\t");

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

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

printf("\n");

}

process = -1;

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

{

if(completed[i] == 0)//if not completed

{

process = i ;

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

{

if(avail[j] < need[i][j])

{

process = -1;

break;

}

}

}

if(process != -1)

break;

}

if(process != -1)

{

printf("\nProcess %d runs to completion!", process + 1);

safeSequence[count] = process + 1;

count++;

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

{

avail[j] += alloc[process][j];

alloc[process][j] = 0;

Max[process][j] = 0;

completed[process] = 1;

}

}

}while(count != p && process != -1);

if(count == p)

{

printf("\nThe system is in a safe state!!\n");

printf("Safe Sequence : < ");

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

printf("%d ", safeSequence[i]);

printf(">\n");

}

else

printf("\nThe system is in an unsafe state!!");

getch();

}

**OUTPUT:**

Enter the no of processes : 5

Enter the no of resources : 3

Enter the Max Matrix for each process :

For process 1 : 1 2 3

For process 2 : 2 3 4

For process 3 : 5 6 7

For process 4 : 8 9 7

For process 5 : 1 5 9

Enter the allocation for each process :

For process 1 : 3 7 9

For process 2 : 5 10 7

For process 3 : 2 8 6

For process 4 : 6 12 5

For process 5 : 9 3 7

Enter the Available Resources : 1 3 5

Max matrix: Allocation matrix:

1 2 3 3 7 9

2 3 4 5 10 7

5 6 7 2 8 6

8 9 7 6 12 5

1 5 9 9 3 7

Process 1 runs to completion!

Max matrix: Allocation matrix:

0 0 0 0 0 0

2 3 4 5 10 7

5 6 7 2 8 6

8 9 7 6 12 5

1 5 9 9 3 7

Process 2 runs to completion!

Max matrix: Allocation matrix:

0 0 0 0 0 0

0 0 0 0 0 0

5 6 7 2 8 6

8 9 7 6 12 5

1 5 9 9 3 7

Process 3 runs to completion!

Max matrix: Allocation matrix:

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

8 9 7 6 12 5

1 5 9 9 3 7

Process 4 runs to completion!

Max matrix: Allocation matrix:

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

1 5 9 9 3 7

Process 5 runs to completion!

The system is in a safe state!!

Safe Sequence : < 1 2 3 4 5 >

**RESULT:**

Thus, a C program to avoid and prevent the deadlock (Banker’s Algorithm) has been executed and required output is obtained.