|  |  |
| --- | --- |
| **EX.NO. : 6**  **DATE:10.04.2021** | **IMPLEMENTATION OF BANKER’S ALGORITHM FOR DEADLOCK AVOIDANCE** |

**Aim:**

To implement banker’s algorithm for deadlock avoidance.

**Algorithm:**

1) Declare the arrays max,need,alloc as 2D arrays and avail, completed safesequence as 1D arrays.

2) Get the no.of. resources,maxmatix and allocation for each process.

3) If available matrix <need don’t implement the process assign process=-1 and break.

4) If not add the process to safesequence array.

5) Repeat this for all process.

6) Print the safesequence array.

**Program:**

#include<stdio.h>

#include<stdlib.h>

int main()

{

intMax[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]);

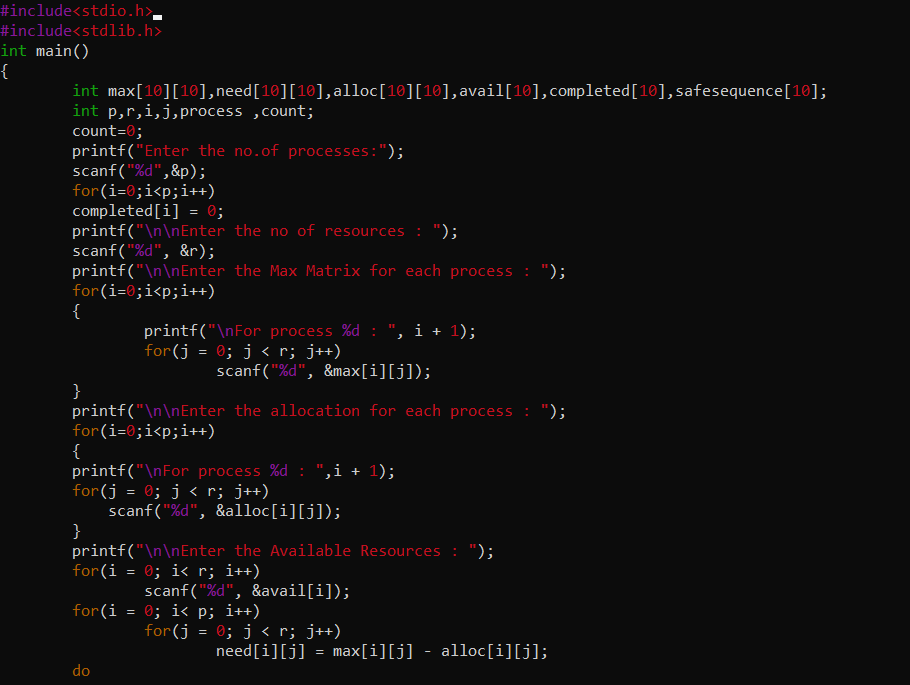
printf(">\n");

}

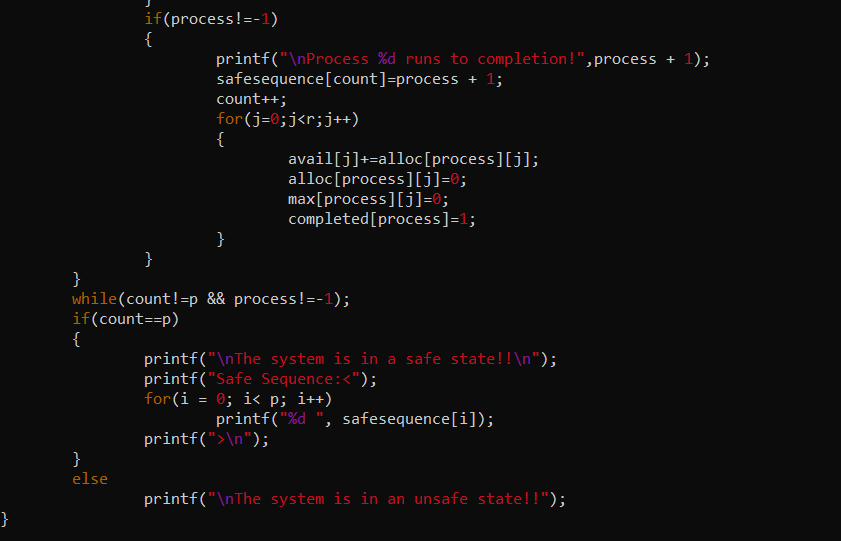
else

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

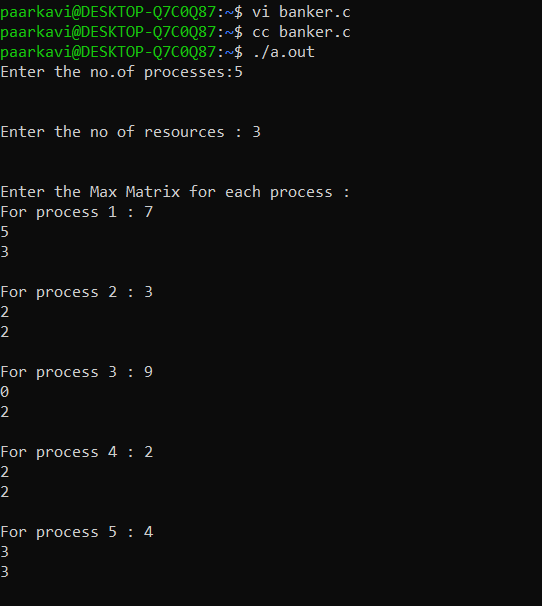
}

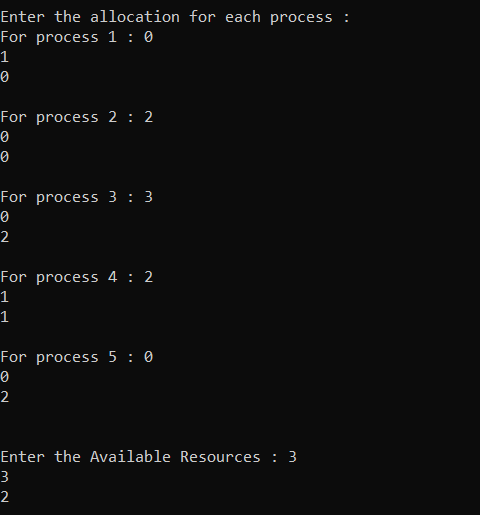


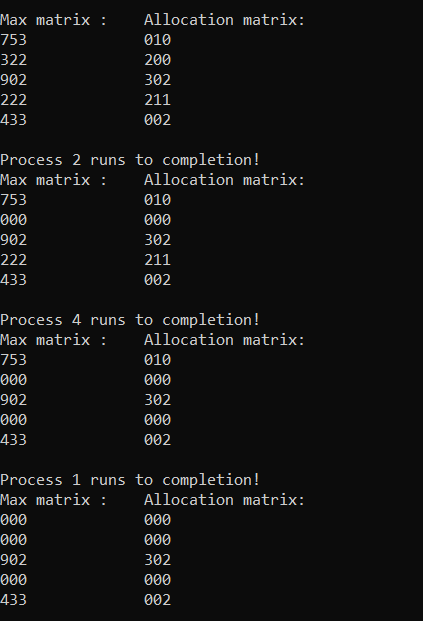


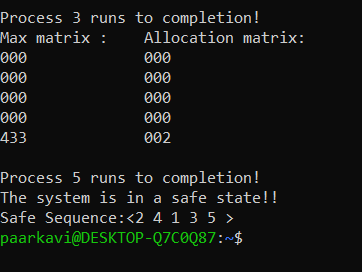


**OUTPUT:**









|  |  |
| --- | --- |
| **Observation(20)** |  |
| **Record(5)** |  |
| **Total(25)** |  |

**Result:**

Thus the implementation of bankers algorithm is verified with the output