Experiment Number: 8

Problem Statement: **Implement Bankers algorithm for a deadlock avoidance and fin out a safe sequence for processes**

Name: Arnav Shah                                                                 Roll No. : 21

Class : AI_C                                                                         Batch : B2

---

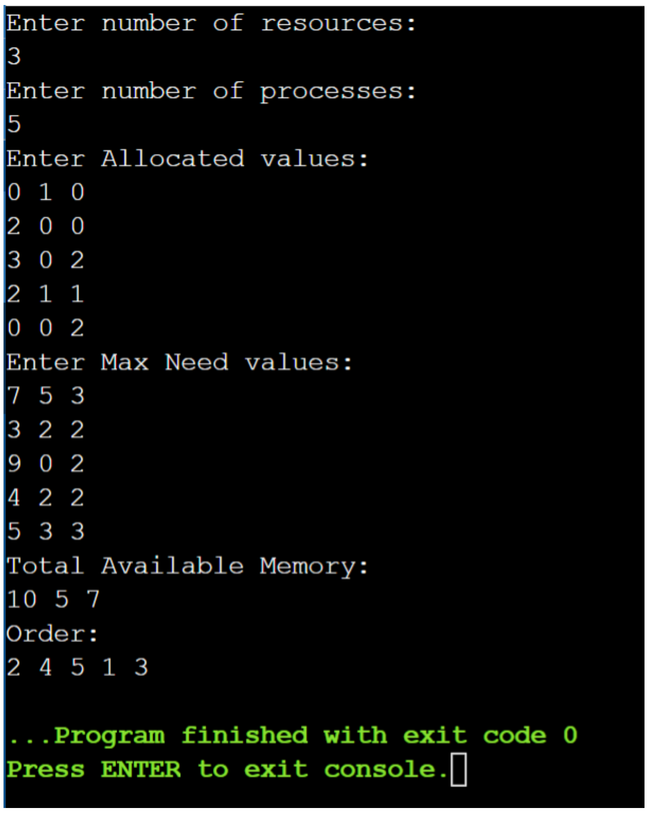
**Code –**

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;
public class bankers {
static int m;
static int n;
static Scanner in=new Scanner(System.in);
public static void input(int[][] matrix,String s){
System.out.println("Enter "+s+" values: ");
for(int i=0;i<m;i++){
for(int j=0;j<n;j++){
matrix[i][j]=in.nextInt();
}
}
}
public static void main(String[] args) {
ArrayList<Integer> arrayList=new ArrayList<>();
System.out.println("Enter number of resources: ");
n=in.nextInt();
System.out.println("Enter number of processes: ");
m=in.nextInt();
int[][] allocate=new int[m][n];
int[][] maxNeed=new int[m][n];
int[][] remNeed=new int[m][n];
int[] flag=new int[m];
Arrays.fill(flag,0);
int[] totalAvailable=new int[n];
int[] totalAllocate=new int[n];
int[] available=new int[n];
input(allocate,"Allocated");
input(maxNeed,"Max Need");
System.out.println("Total Available Memory: ");
for(int i=0;i<n;i++){
totalAvailable[i]=in.nextInt();
}
```

```
for(int i=0;i<m;i++){
for(int j=0;j<n;j++){
remNeed[i][j]=maxNeed[i][j]-allocate[i][j];
}
}
// System.out.println(Arrays.deepToString(remNeed));
int sum=0;
for(int i=0;i<n;i++){
sum=0;
for(int j=0;j<m;j++){
sum+=allocate[j][i];
}
totalAllocate[i]=sum;
}
for(int i=0;i<n;i++){
available[i]=totalAvailable[i]-totalAllocate[i];
}
// System.out.println(Arrays.toString(available));
int count=0;
int release=0;
boolean flg=true;
while(flg){
for(int i=0;i<m;i++){
for(int j=0;j<n;j++){
if(remNeed[i][j]<=available[j] && flag[i]==0){
count++;
if(count==n){
count=0;
flag[i]=1;
for(int k=0;k<n;k++){
available[k]=available[k]+allocate[i][k];
}
arrayList.add(i+1);
}
}
else{
release++;
if(release==m){
flg=false;
}
break;
}
}
}
}
```

```
if(arrayList.isEmpty()){
System.out.println("Deadlock occurs.");
}
else{
System.out.println("Order: ");
for(int i: arrayList){
System.out.print(i+" ");
}
}
}
}
```

Output –



```
Enter number of resources:
3
Enter number of processes:
5
Enter Allocated values:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter Max Need values:
7 5 3
3 2 2
9 0 2
4 2 2
5 3 3
Total Available Memory:
10 5 7
Order:
2 4 5 1 3

...Program finished with exit code 0
Press ENTER to exit console.
```