1. a. This is a resource livelock. Here the stations are still transmitting frames and they detect collision and try to transmit the frame again. The stations are still working and have not come to a standstill. But it still can't progress ahead due to collisions and retransmission of frames. Hence it's a system livelock and not a resource deadlock as none of the systems are waiting infinitely for any resource.
   b. If the two systems send frames after different time intervals (randomly) instead of the same time intervals they are always trying for then this collision can be prevented.
   c. Yes starvation occurs in this case as the stations are continuously trying to send frames after collision at the same time without any progress of transmitting it successfully.

2. It's a communication deadlock. In communication deadlocks, processes wait to communicate with other processes among a set of processes. A waiting process can unblock on receiving a communication from one of these processes. A set of processes is communication-deadlocked if each process in the set is waiting to communicate with another process in the set and no process ever initiates any further communication until it receives the communication for which it is waiting.
   Here all the four cars aren't waiting for any resource but they are waiting for message from the two neighbor cars. Any of the cars doesn't possess the resources but they are still deadlocked and can be solved by different timeouts.

3. Process P2 gets the available resources and completes its tasks and then releases its resources. Availability matrix then is (0,2,0,3,1).
   P3 then gets the available resources and completes its tasks and then releases its resources. Availability matrix then is (0,2,0,3,2).
   Now P1 and P4 are deadlocked as P1 requires 1 quantity of RS1 which is not available and P4 requires 1 of RS3. Both cannot get their resources. Hence P1 and P4 are deadlocked.

4. Solution1: If we add the Allocated Resources Matrix and the Available Matrix we get the total number of resources. Hence the total number of resources available are (5, 2, 4+x, 5, 2). The maximum resources that a process can demand should be always less than the total number of resources. Here process A demands maximum 3units of RS5 which is less than total number of RS5 in the system. So this will never lead to safe state for any value of x.
   Solution2: We first created required matrix as
   R: A 0 1 0 0 2
   B 0 2 1 0 0
   C 1 0 3 0 0
   D 0 0 1 1 1
   Allocated:
   A 1 0 2 1 1
   B 2 0 1 1 0
   C 1 1 0 1 0
   D 1 1 1 1 0

   A: 0 0 x 1 1

If we consider process D then x = 1; we process D and then complete the task and release the resources.
A: 11221
Now if we had x =2 we could get A:11321 and we can consider process C and then complete the task and release the resources.
A:22331
Now we consider process B and then complete the task and release the resources.
A:42441

Still we couldn't complete process A as it requires 2 units of resource 5 which is not available. Hence no safe state and it is not dependent on x only.

5.

```
/*Bankers Algorithm Simulation: Considering the Resource Assigned matrix, Resources still to be
assigned matrix,total resources,
 * total clients,and total existing resources matrix is given in an input.txt file.The input file is read in the
class ReadFile
 * and the availableResource matrix is computed. all the matrices are used in the bankers algorithm to
compute the safe state.
 * Driver program cycles through each of the client and asks for their request for resources. If its safe to
grant them resources
 * then they are granted and available matrix is updated accordingly. If it leads to unsafe state then the
resources are not
 * granted and corresponding message is shown.The output is shown on console as well as saved in
output.txt file.
 *
 */
import java.io.IOException;
import java.util.Arrays;

public class BankersAlgorithm {

	private int resources = 0, client = 0;
	private int availableRes[] = null;
	private int resAssigned[][] = null;
	private int resStilltoAssign[][] = null;

	public int bankersAlgorithm(ReadFile readData, int required[],int clientNumber) throws
IOException {

		client			= readData.getClient();
		resources		= readData.getResources();
		availableRes		= new int[resources];
		resAssigned		= new int[client][resources];
		resStilltoAssign	= new int[client][resources];
		resAssigned			= readData.getResAssigned();
		System.arraycopy(readData.getAvailable(), 0,availableRes ,
0,readData.getAvailable().length );
		for(int k = 0;k<client;k++){
			resStilltoAssign[k] = Arrays.copyOf(readData.getResStilltoAssign()[k],
readData.getResStilltoAssign().length);
		}

		int terminated[] = new int[client];
		int safe = 0;
		int unsafe = 0;
		int j;

		for(int i = 0;i<client;i++){
			terminated[i] = 0;
		}

		for(int k = 0;k<resources;k++){
			if(resStilltoAssign[clientNumber][k] >= required[k]){
				if(availableRes[k] >= required[k]){
					availableRes[k]= availableRes[k] - required[k];
					resStilltoAssign[clientNumber][k]-=required[k];
				}
				else{
					System.out.println("Requested number of resources is more
than available");
					return 1;

				}
			}
			else{
				System.out.println("Requested number of resources is more "
						+ "than maximum limit allocated for client:"+
(clientNumber + 1));
				return 2;
			}
		}
		System.out.println("Checking to see if its a safe state!!!");
		while(safe != client && unsafe != client){
			unsafe++;
			for(int i = 0;i<client;i++){
				if(terminated[i] == 0){
					for(j = 0;j<resources;j++){
						if(resStilltoAssign[i][j] > availableRes[j] ){
							break;
						}
					}
					if(j == resources){
```

```
                                                System.out.println("Processing client:"+ (i+1));
                                                terminated[i] = 1;
                                                safe++;
                                                for(int k = 0;k<resources;k++){
                                                        availableRes[k]+=resAssigned[i][k];
                                                }
                                        }
                                }
                        }
                }
                if(safe == client){
                        System.out.println("Its a safe state!!!");
                        int availableFinal[] = readData.getAvailable();
                        int resStilltoAssignFinal[][] = readData.getResStilltoAssign();
                        System.out.println("Now available resources:");
                        for(int m = 0;m<resources;m++){
                                availableFinal[m]= availableFinal[m] - required[m];
                                resStilltoAssignFinal[clientNumber][m]-=required[m];
                                System.out.print(availableFinal[m] + " ");
                        }
                        System.out.println("\n");
                        return 3;
                }
                else{
                        System.out.println("Deadlock!!!Its an unsafe state");
                        return 4;
                }

        }
}


import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.StringTokenizer;

public class ReadFile {

        private int resources = 0, client = 0;
        private int totResource[] = null;
        private int available[] = null;
        private int possesedRes[] = null;
        private int resAssigned[][] = null;
        private int resStilltoAssign[][] = null;

        String line = null;
        int i = 0;
```

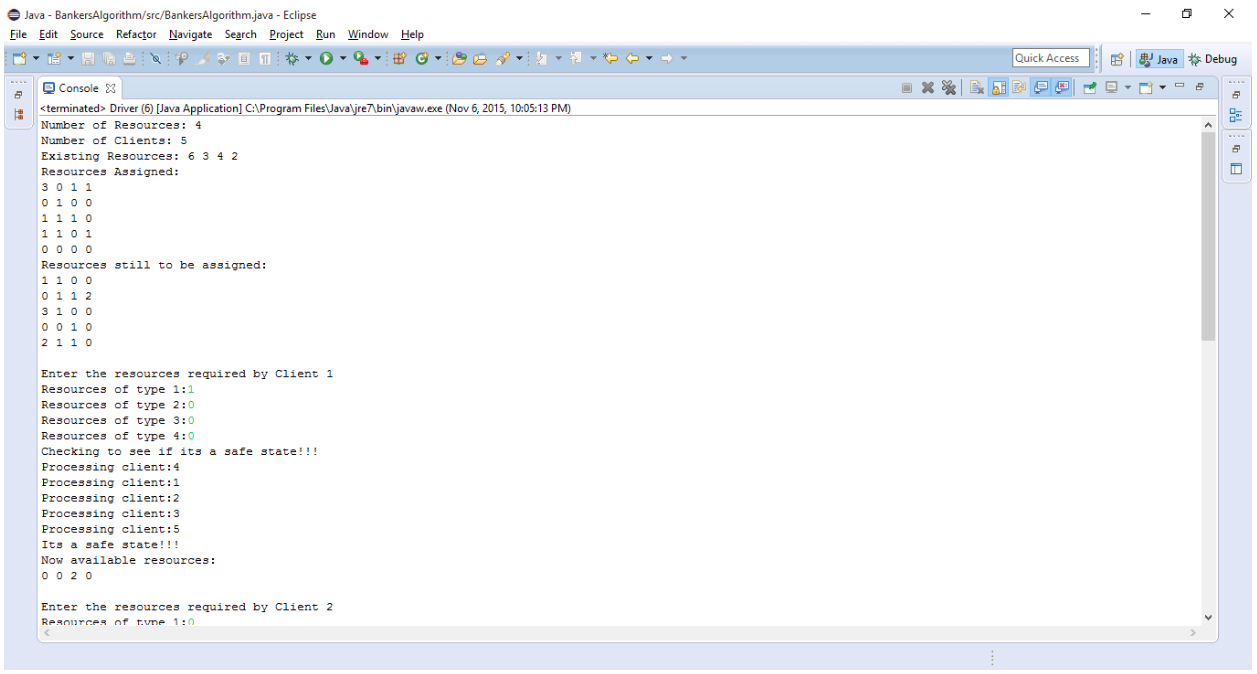
```
public void readFile(){

        try {
                // FileReader reads text files in the default encoding.
                FileReader fileReader = new FileReader("input.txt");

                // Always wrap FileReader in BufferedReader.
                BufferedReader bufferedReader =
                                new BufferedReader(fileReader);

                while((line = bufferedReader.readLine()) != null) {
                        StringTokenizer st = new StringTokenizer(line);
                        String token = null;
                        int j = 0;
                        while (st.hasMoreTokens()) {
                                token = st.nextToken();
                                System.out.print(token +" ");

                                if(i==2 && j > 1){
                                        totResource[j-2] = Integer.parseInt(token);
                                }
                                if(i>3 && i<= client + 3){
                                        resAssigned[i-4][j] = Integer.parseInt(token);

                                }
                                if(i > client + 4 && i<= 2*client + 4){
                                        resStilltoAssign[i- client - 5][j] = Integer.parseInt(token);

                                }
                                j++;
                        }
                        System.out.print("\n");
                        if(i==0){
                                resources = Integer.parseInt(token);
                        }
                        else if(i==1){
                                client = Integer.parseInt(token);
                                totResource = new int[resources];
                                available = new int[resources];
                                possesedRes = new int[resources];
                                resAssigned = new int[client][resources];
                                resStilltoAssign = new int[client][resources];

                        }
                        i++;
                }
        }
```

```
		catch(IOException ex) {
			System.out.println("Error reading file ");
			ex.printStackTrace();

		}

		for(int k = 0; k<client;k++){
			for(int m = 0;m<resources;m++){
				possesedRes[m]+= resAssigned[k][m];
			}
		}

		for(int m = 0;m<resources;m++){
			available[m] = totResource[m] - possesedRes[m];
		}

	}

	public int getResources() {
		return resources;
	}

	public int getClient() {
		return client;
	}

	public int[] getTotResource() {
		return totResource;
	}

	public int[] getAvailable() {
		return available;
	}

	public int[] getPossesedRes() {
		return possesedRes;
	}

	public int[][] getResAssigned() {
		return resAssigned;
	}

	public int[][] getResStilltoAssign() {
		return resStilltoAssign;
	}
}

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class Driver {

    public static void main(String[] args) throws IOException {

        ReadFile readData = new ReadFile();//Read file
        readData.readFile();
        int client                  = readData.getClient();
        int resources     = readData.getResources();


        PrintWriter out = new PrintWriter(new FileWriter("output.txt"));
        Scanner in = new Scanner(System.in);

        int required[] = new int[resources];

        //ask request from each client and show whether its granted or not on console as well
as output file
        for(int i = 0;i<client;i++){
            System.out.println("Enter the resources required by Client "+ (i+1));
            out.print("List of resources requested by Client ");
            out.print((i+1));
            out.print(" are:");
            for(int j =0;j<resources;j++){
                required[j] = 0;
                System.out.print("Resources of type "+ (j+1) + ":");
                required[j]  = in.nextInt();
                out.print(required[j]);
            }
            BankersAlgorithm b = new BankersAlgorithm();
            out.println();
            int returnValue = b.bankersAlgorithm(readData,required,i);
            if(returnValue == 1){
                out.print("Requested number of resources are more than available");
            }
            else if(returnValue == 2){
                out.print("Requested number of resources is more "
                        + "than maximum limit allocated for client");

            }
            else if(returnValue == 3){
                out.println("Its a safe state!!!");
                out.print("Now Available Resources:");
                int availableFinal[] = readData.getAvailable();
                for(int m = 0;m<resources;m++){
                    out.print(availableFinal[m]);
                }
            }
            else if(returnValue == 4){
                out.print("Its an unsafe state!!!");
            }
            out.println("\n");

        }
        out.close();
    }
}
```

Input file:

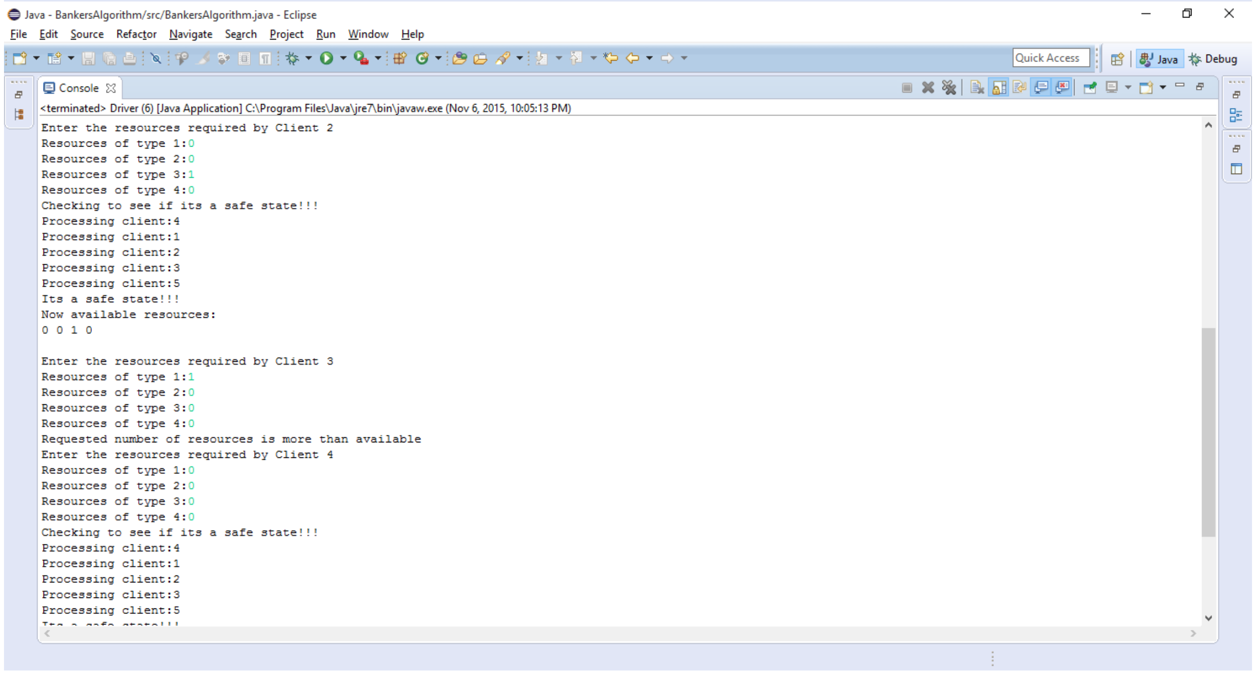
```
Number of Resources: 4
Number of Clients: 5
Existing Resources: 6 3 4 2
Resources Assigned:
3 0 1 1
0 1 0 0
1 1 1 0
1 1 0 1
0 0 0 0
Resources still to be assigned:
1 1 0 0
0 1 1 2
3 1 0 0
0 0 1 0
2 1 1 0
```

Output:


Java - BankersAlgorithm/src/BankersAlgorithm.java - Eclipse

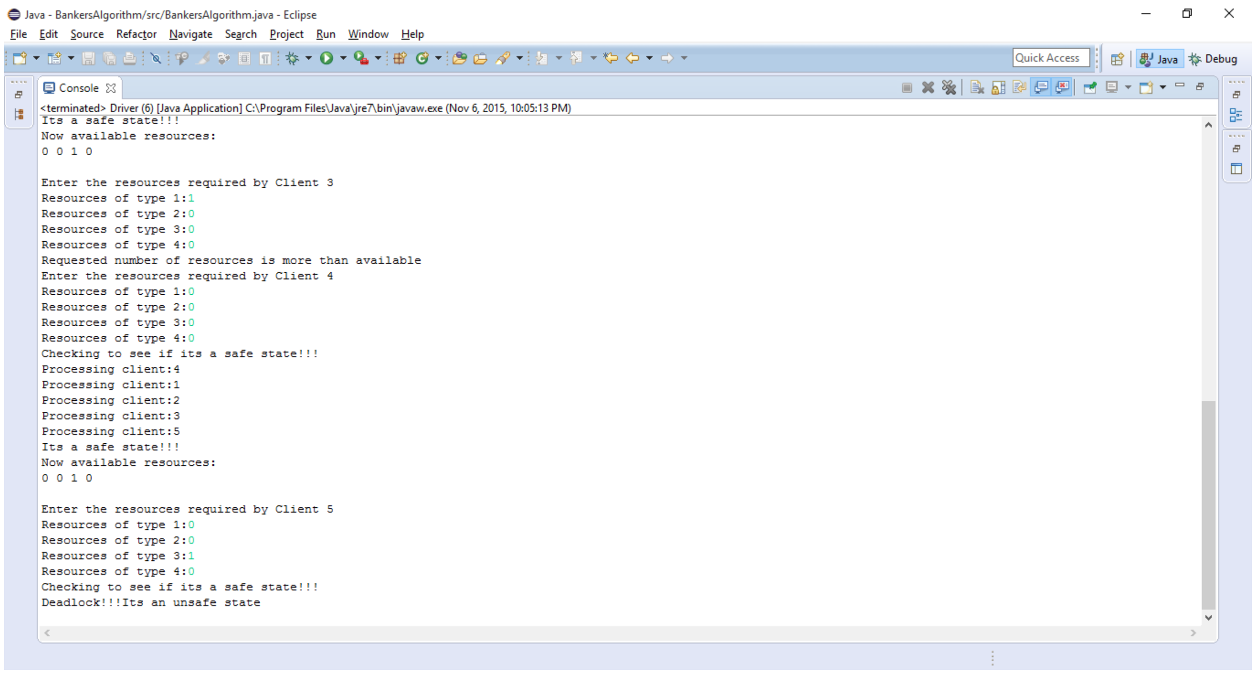
<terminated> Driver (6) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Nov 6, 2015, 10:05:13 PM)

```
Number of Resources: 4
Number of Clients: 5
Existing Resources: 6 3 4 2
Resources Assigned:
3 0 1 1
0 1 0 0
1 1 1 0
1 1 0 1
0 0 0 0
Resources still to be assigned:
1 1 0 0
0 1 1 2
3 1 0 0
0 0 1 0
2 1 1 0

Enter the resources required by Client 1
Resources of type 1:1
Resources of type 2:0
Resources of type 3:0
Resources of type 4:0
Checking to see if its a safe state!!!
Processing client:4
Processing client:1
Processing client:2
Processing client:3
Processing client:5
Its a safe state!!!
Now available resources:
0 0 2 0

Enter the resources required by Client 2
```


Java - BankersAlgorithm/src/BankersAlgorithm.java - Eclipse

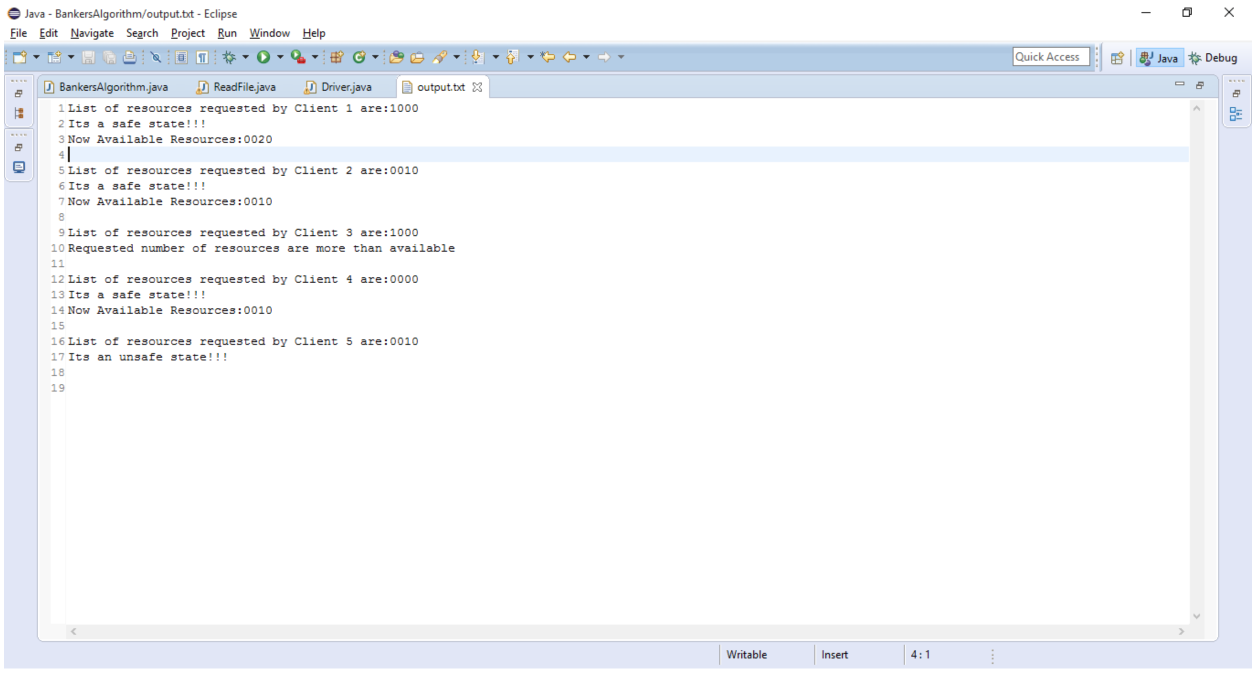
<terminated> Driver (6) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Nov 6, 2015, 10:05:13 PM)

```
Enter the resources required by Client 2
Resources of type 1:0
Resources of type 2:0
Resources of type 3:1
Resources of type 4:0
Checking to see if its a safe state!!!
Processing client:4
Processing client:1
Processing client:2
Processing client:3
Processing client:5
Its a safe state!!!
Now available resources:
0 0 1 0

Enter the resources required by Client 3
Resources of type 1:1
Resources of type 2:0
Resources of type 3:0
Resources of type 4:0
Requested number of resources is more than available
Enter the resources required by Client 4
Resources of type 1:0
Resources of type 2:0
Resources of type 3:0
Resources of type 4:0
Checking to see if its a safe state!!!
Processing client:4
Processing client:1
Processing client:2
Processing client:3
Processing client:5
```


Java - BankersAlgorithm/src/BankersAlgorithm.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Console

```
<terminated> Driver (6) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Nov 6, 2015, 10:05:13 PM)
Its a safe state!!!
Now available resources:
0 0 1 0

Enter the resources required by Client 3
Resources of type 1:1
Resources of type 2:0
Resources of type 3:0
Resources of type 4:0
Requested number of resources is more than available
Enter the resources required by Client 4
Resources of type 1:0
Resources of type 2:0
Resources of type 3:0
Resources of type 4:0
Checking to see if its a safe state!!!
Processing client:4
Processing client:1
Processing client:2
Processing client:3
Processing client:5
Its a safe state!!!
Now available resources:
0 0 1 0

Enter the resources required by Client 5
Resources of type 1:0
Resources of type 2:0
Resources of type 3:1
Resources of type 4:0
Checking to see if its a safe state!!!
Deadlock!!!Its an unsafe state
```

Output file(output.txt)



```
List of resources requested by Client 1 are:1000
Its a safe state!!!
Now Available Resources:0020

List of resources requested by Client 2 are:0010
Its a safe state!!!
Now Available Resources:0010

List of resources requested by Client 3 are:1000
Requested number of resources are more than available

List of resources requested by Client 4 are:0000
Its a safe state!!!
Now Available Resources:0010

List of resources requested by Client 5 are:0010
Its an unsafe state!!!
```