Shilpita Roy

COEN 283 Fall 2015 ASSIGNMENT 3 Deadlocks

1) Local Area Networks utilize a media access method called CSMA/CD, in which stations sharing a bus can sense the medium and detect transmissions as well as collisions. In the Ethernet protocol, stations requesting the shared channel do not transmit frames if they sense the medium is busy. When such transmission has terminated, waiting stations each transmit their frames. Two frames that are transmitted at the same time will collide. If stations immediately and repeatedly retransmit after collision detection, they will continue to collide indefinitely.

(a) Is this a resource deadlock or a livelock?
Ans: This is **Livelock** as no station is able to transmit without collision.

(b) Can you suggest a solution to this anomaly?
Ans: The two stations should set up timers with different time stamp so that they are not transmitting at the same time thus avoiding collision. There needs to be a protocol between the stations so the transmission should be at random time.

(c) Can starvation occur with this scenario?
Ans: No, since both the stations are using the channel to transmit their frames and they are not short of resource. But they are unable to complete transmission due to the collision.

2) Suppose four cars each approach an intersection from four different directions simultaneously. Each corner of the intersection has a stop sign. Assume that traffic regulations require that when two cars approach adjacent stop signs at the same time, the car on the left must yield to the car on the right. Thus, as four cars each drive up to their individual stop signs, each waits (indefinitely) for the car on the left to proceed. Is this anomaly a communication deadlock? Is it a resource deadlock?

Ans: This is communication deadlock as it's an anomaly of co-operative synchronization. The car to the left is waiting for the cars to the right to move. Thus, the processes (cars) are waiting for other process for co-operation.

In communication deadlock there are no classical resources involved. Here the path is open to the cars but they need co-ordination for moving forward. This deadlock can be resolved by setting up random waiting time for each car.

3) Consider the following state of a system with four processes, P1, P2, P3, and P4, and five types of resources, RS1, RS2, RS3, RS4, and RS5:

$$C = \begin{bmatrix} 0 & 1 & 1 & 1 & 2 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 1 & 0 & 2 & 1 \\ 0 & 1 & 0 & 2 & 1 \\ 0 & 2 & 0 & 3 & 1 \\ 0 & 2 & 1 & 1 & 0 \end{bmatrix} \qquad \begin{array}{l} E = (24144) \\ A = (01021) \end{array}$$

Show that there is a deadlock in the system. Identify the processes that are deadlocked.

Ans: The Process P1 and P4 are deadlocked.
We can complete P2 and then P3. Thus the available resource matrix will become
A = (0 2 0 3 2).
But this will not be enough to fulfill requests of either P1 or P4.
P1 requests for 1instance of R1 and is holding the only 1 instance of R3 while P4 is holding all 2 instances of R1 and requesting 1 of R3 that P1 is holding.

4) A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows:

	Allocated	*Maximum*	*Available*
Process A	1 0 2 1 1	1 1 2 1 3	0 0 x 1 1
Process B	2 0 1 1 0	2 2 2 1 0	
Process C	1 1 0 1 0	2 1 3 1 0	
Process D	1 1 1 1 0	1 1 2 2 1	

What is the smallest value of x for which this is a safe state?

Ans: E= [5 2 (4+x) 5 2] . For any value of X the given situation is not a safe state as the maximum requested resource R5 for process A is 3 while system has only 2 resources of R5.

But if the maximum requirement of process A was [1 1 2 1 1] thus maximum matrix becoming

	Allocated	Maximum	Need
Process A	1 0 2 1 1	1 1 2 1 **1**	0 1 0 0 0
Process B	2 0 1 10	2 2 2 1 0	0 2 1 0 0
Process C	1 1 0 1 0	2 1 3 1 0	1 0 3 0 0
Process D	1 1 1 1 0	1 1 2 2 1	0 0 1 1 0

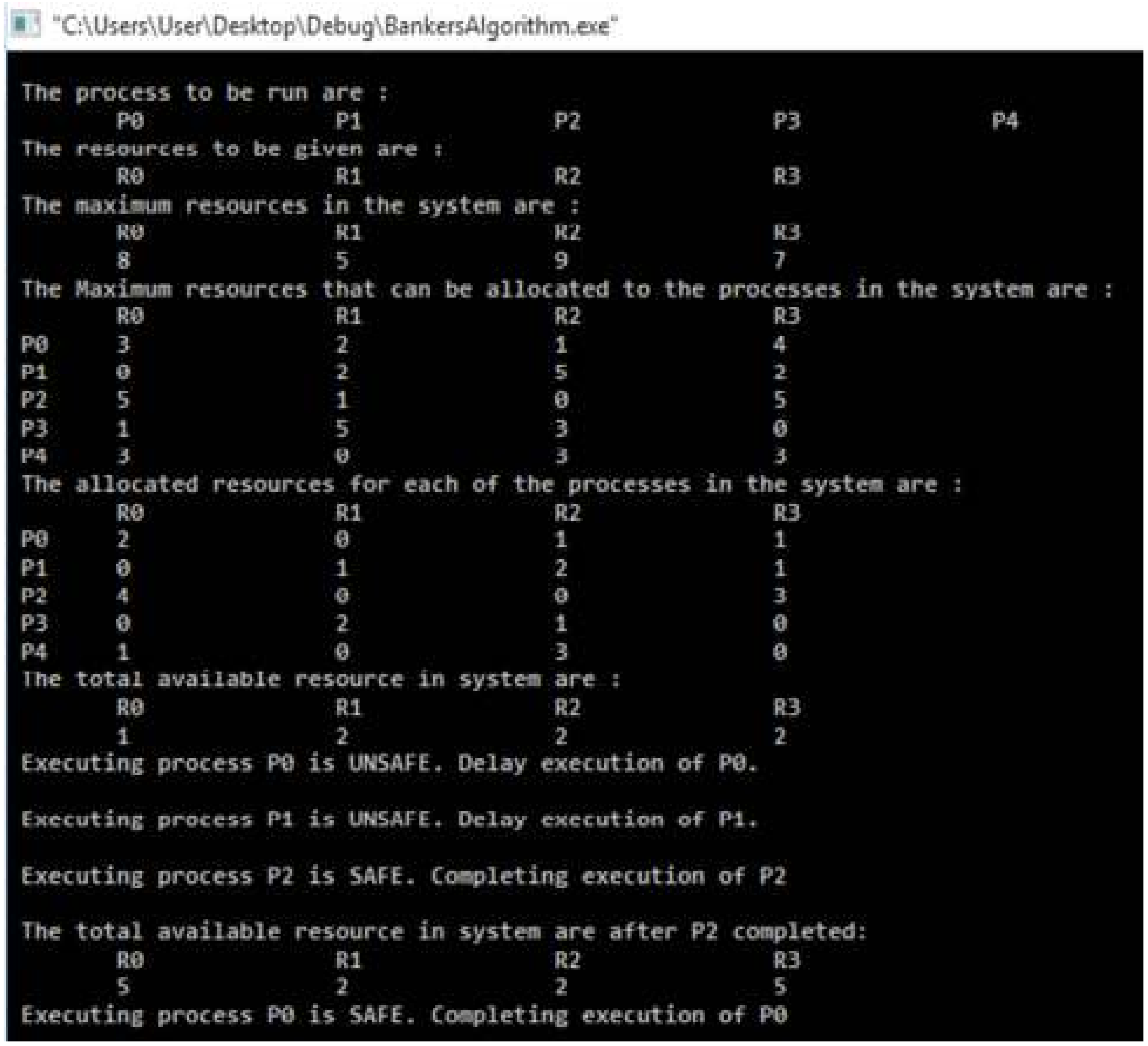
In this case if X=0 or X= 1 then there is deadlock (if x=1 on D can run). If X=2 then A=(0 0 1 1 1) and the order of execution can be D -> A -> B ->C.

5) Program a simulation of the banker's algorithm. Your program should cycle through each of the bank clients asking for a request and evaluating whether it is safe or unsafe. Output a log of requests and decisions to a file.
Soutce code:

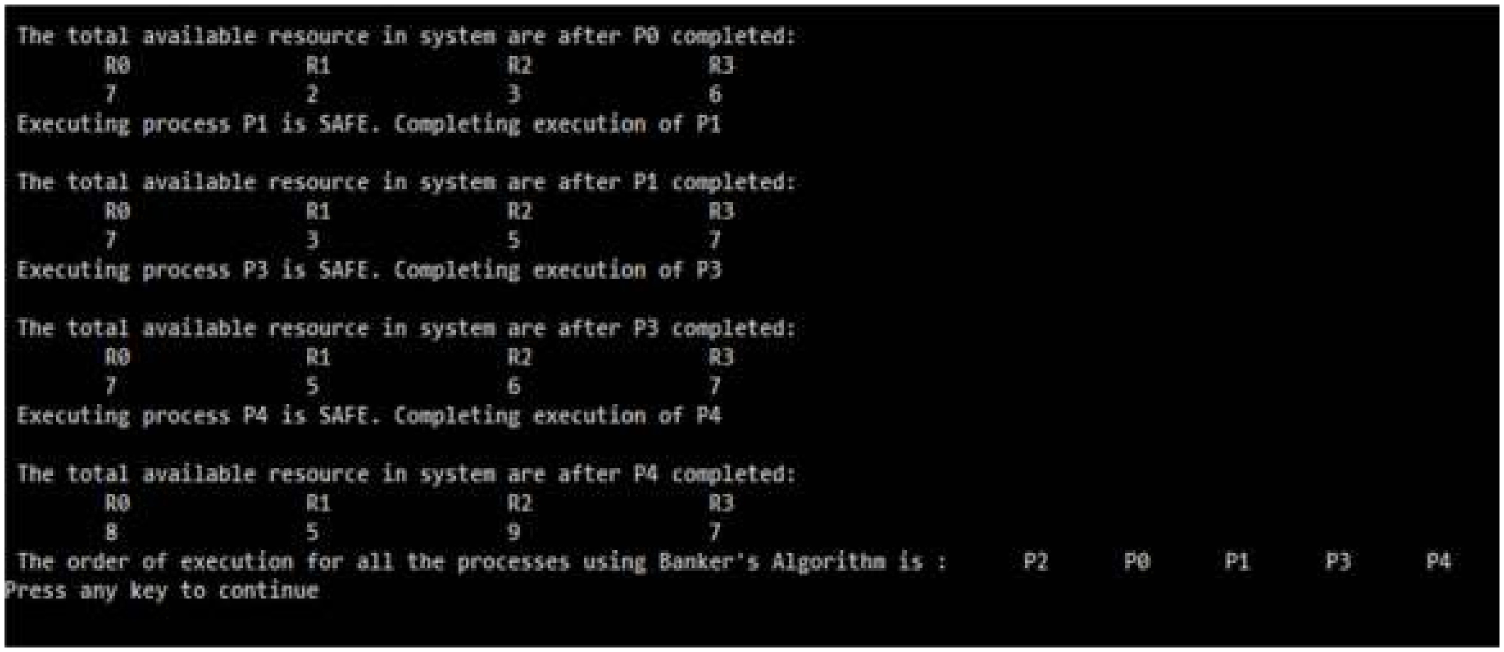
```
#include <stdio.h>

#define Max_Resource_no 4
#define Max_process_no 5


void Print_Resource(int resource[Max_Resource_no])
{
   int i;
   for (i=0 ; i<Max_Resource_no ;i++ )
        printf ("\tR%d\t", i);
        printf ("\n");
   for (i=0 ; i<Max_Resource_no ;i++ )
        printf ("\t%d \t",resource[i]);
}

/*To print matrices*/
void Print_Matrices(int a[Max_process_no][Max_Resource_no])
{
   int i, j;
   for (i=0 ; i<Max_Resource_no ;i++ )
        printf ("\tR%d\t",i);
   for (i=0 ; i<Max_process_no ;i++ )
   {
      printf ("\n P%d \t",i);
      for (j=0 ; j<Max_Resource_no ;j++ )
      {
         printf("%d \t\t",a[i][j]);
      }
   }
}

void GetAvailResource(int current_loan[Max_process_no][Max_Resource_no], int
available_resource[Max_Resource_no],int total_resource[Max_Resource_no])
{
   int i , j , sum =0;
   for (i=0 ; i < Max_Resource_no ;i++ )
   {
                 sum=0;
     for (j=0 ; j< Max_process_no ;j++ )
      {
         sum+= current_loan[j][i];
      }

      available_resource[i] =  total_resource[i] - sum;
   }
}

void BankingResource(int available_resource[Max_Resource_no],int
current_loan[Max_process_no][Max_Resource_no],int
max_loan[Max_process_no][Max_Resource_no])
{

         int i , j ,  exec_flag ,safe_flag;
         int count = Max_process_no +1;

         int pexec_order[Max_process_no]; int k =0 ;
         int run_process[Max_process_no] ={1,1,1,1,1};

         while(count != 0)

   {
                 safe_flag =0;

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

                  {
                           if (run_process[i] ==1)
                           {
                                   exec_flag =1;
                                                   for(j=0 ; j< Max_Resource_no ; j++)
                                                           {
                                                                   if (available_resource[j] < (max_loan[i][j] -
current_loan[i][j]))

                                                                   {
                                                                           printf("\n Executing process P%d is
UNSAFE. Delay execution of P%d.\n",i,i);
                                                                           exec_flag =0;
```

```
                                                                    safe_flag =0;
                                                                    break;
                                                            }
                                                    }

                                    if (exec_flag == 1)
                                    {
                                      run_process[i] =0;
                                      printf("\n Executing process P%d is SAFE. Completing
execution of P%d\n",i,i);

                                            pexec_order[k++]=i;

                                      for (j=0 ; j< Max_Resource_no ; j++)
                                      {
                                            available_resource[j] = available_resource[j]+
current_loan [i][j];
                                      }
                                      printf("\n The total available resource in system are after
P%d completed: \n",i);
                                      Print_Resource(available_resource);
                                      safe_flag =1;
                                      count--;
                                      //printf("\n%d",count);
                                      break;

                                    }
                      }
                }
     if(count == 1) break;
       }
  printf("\n The order of execution for all the processes using Banker's Algorithm is :\t");
  for(i=0; i< Max_process_no; i++)
              printf (" P%d\t",pexec_order[i]);
  printf("\n");
}

int main()
{
   int total_resource[Max_Resource_no] = {8, 5, 9, 7}; //Effective resource vector
   int available_resource[Max_Resource_no]; // Available resource vector
   /*Number of resources held by process*/
   int current_loan[Max_process_no][Max_Resource_no] = {

         {2, 0, 1, 1},{0, 1, 2, 1},{4, 0, 0, 3},{0, 2, 1, 0},{1, 0, 3, 0}
                                                     };

   /*Maximum resouces need by the processes to complete */
   int max_loan[Max_process_no][Max_Resource_no]= {
         {3, 2, 1, 4},{ 0, 2, 5, 2},{ 5, 1, 0, 5},{1, 5, 3, 0},{3, 0, 3, 3}
                                                 };
     int i;    // index for loops and counter for running process

  // int p_exec_order[Max_process_no];

   printf("\n The process to be run are : \n");
   for (i=0 ; i<Max_process_no ;i++ )
         printf ("\tP%d \t", i);

   printf("\n The resources to be given are : \n");
   for (i=0 ; i<Max_Resource_no ;i++ )
         printf ("\tR%d \t", i);

     printf("\n The maximum resources in the system are : \n");
   Print_Resource(total_resource);

   printf("\n The Maximum resources that can be allocated to the processes in the system are :
\n");
   Print_Matrices(max_loan);

   printf("\n The allocated resources for each of the processes in the system are : \n");
   Print_Matrices(current_loan);

   printf("\n The total available resource in system are : \n");
         GetAvailResource(current_loan,available_resource,total_resource);
   Print_Resource(available_resource);

   BankingResource(available_resource,current_loan,max_loan);

   return 0;
}
```

Output :



```
"C:\Users\User\Desktop\Debug\BankersAlgorithm.exe"

The process to be run are :
        P0              P1              P2              P3              P4
The resources to be given are :
        R0              R1              R2              R3
The maximum resources in the system are :
        R0              R1              R2              R3
        8               5               9               7
The Maximum resources that can be allocated to the processes in the system are :
        R0              R1              R2              R3
P0      3               2               1               4
P1      0               2               5               2
P2      5               1               0               5
P3      1               5               3               0
P4      3               0               3               3
The allocated resources for each of the processes in the system are :
        R0              R1              R2              R3
P0      2               0               1               1
P1      0               1               2               1
P2      4               0               0               3
P3      0               2               1               0
P4      1               0               3               0
The total available resource in system are :
        R0              R1              R2              R3
        1               2               2               2
Executing process P0 is UNSAFE. Delay execution of P0.

Executing process P1 is UNSAFE. Delay execution of P1.

Executing process P2 is SAFE. Completing execution of P2

The total available resource in system are after P2 completed:
        R0              R1              R2              R3
        5               2               2               5
Executing process P0 is SAFE. Completing execution of P0
```



```
The total available resource in system are after P0 completed:
        R0              R1              R2              R3
        7               2               3               6
Executing process P1 is SAFE. Completing execution of P1

The total available resource in system are after P1 completed:
        R0              R1              R2              R3
        7               3               5               7
Executing process P3 is SAFE. Completing execution of P3

The total available resource in system are after P3 completed:
        R0              R1              R2              R3
        7               5               6               7
Executing process P4 is SAFE. Completing execution of P4

The total available resource in system are after P4 completed:
        R0              R1              R2              R3
        8               5               9               7
The order of execution for all the processes using Banker's Algorithm is :      P2      P0      P1      P3      P4
Press any key to continue
```