CP 386, Fall 2019
Assignment 2 (10% of the final grade)
(due Thursday, November 7, at 11:00 pm)

Task

Your task is to implement a deadlock detection and correction algorithm for an OS as a C function (see **Implementation** at the end of this document for details). It will process the "current state" of a system, detect if there is a deadlock, and act correspondingly to the result of the detection (as described in lecture notes and the textbook).

Preliminary notation (as given in lecture notes):
R_i will denote resource request vector for process i;
A_i will denote current resource allocation vector for process i;
U will denote unallocated (available) resource vector;

Input

The function takes 1 (one) parameter – a file pointer pointing to the data describing the current state of the system. The file contains several lines:
First line

```
n m
```

contains two integers: n - number of processes in the system (assume processes are numbered from 1 to n), and m - number of resources (assume resources are numbered from 1 to m). This line will be followed by $2n + 1$ lines, each containing m integer numbers:
$R_{11}, R_{12}, ..., R_{1m}$
$R_{21}, R_{22}, ..., R_{2m}$
...
$R_{n1}, R_{n2}, ..., R_{nm}$
$A_{11}, A_{12}, ..., A_{1m}$
$A_{21}, A_{22}, ..., A_{2m}$
...
$A_{n1}, A_{n2}, ..., A_{nm}$
$U_1, U_2, ..., U_m$

For example the input

```
3 5
0 1 0 0 0
1 0 0 0 0
0 0 0 0 0
```

```
1 0 0 1 0
0 2 1 0 0
0 1 1 0 1
0 0 0 0 0
```

corresponds to the example on page 54 of Unit04 of the lecture notes (system is not deadlocked).

The input

```
3 5
0 1 0 0 0
0 0 0 0 1
0 1 0 0 0
1 0 0 0 0
0 2 0 0 0
0 1 1 0 1
0 0 1 1 0
```

corresponds to the example on page 53 of Unit04 of the lecture notes (system is deadlocked).

Your function will have to detect deadlock based on provided input.

If the system is not deadlocked your function will print the order of completion of processes. For the first example above this will be

```
3 1 2
```

Note: If more than one process can complete the execution at any time, print them in the decreasing order of process number.

If the system is deadlocked you function will have to find a possible resolution for the deadlock: i.e. it has to find one or more processes which needs to be terminated to resolve the deadlock, and show the order of completion for the remaining processes. The output in this case will contain three lines:
first line contains process numbers for processes involved in the deadlock,
second line contains process numbers for processes to be terminated,
third line contains the order of completion of the remaining processes.
For the second example above this will be:

```
1 2 3
3
1 2
```

The decision which of the processes involved into deadlock needs to be terminated is based on the following very simple rule: terminate the process with the **largest process number** whose termination will resolve the deadlock.

Note: There might be more than one deadlock cycle in the system. It is possible that more than one porcess needs to be terminated in order to resolve the deadlock.

Implementation

You will be given the following files :

- makefile
- main.c
- detector.h
- **detector.c**

Do not edit any of the files except for detector.c where your code goes. The marker should be able to use an entirely different main.c and get the same results as you. You are encouraged to add additional .c and .h files as you desire to implement and useful data structures as long as your code still compiles using the makefile. If you reuse code from previous years you must make this obvious in the header of these files.

Marking

Your code will be tested against several test cases. Some of these will be provided on MLS prior to the due date and some will not. The expectation is that if your code passes the provided tests cases it should pass the additional ones, so if you only pass the provided test cases you will pass the assignment (although not with flying colors). Expect the additional tests cases to cover less obvious edge cases.
Public test cases along with the expected output will be posted on MLS.

What to Submit

Upload **1 zip file** to the MLS dropbox named **Laurier ID #-CP386-A2.zip** . Example if your student ID # is 1723571113 you would submit **1723571113-CP386-A2.zip**

Failure to name your submission correctly will result in a mark of zero.