In this assignment, you will implement the deadlock detection algorithm with multiple resources of each type. The algorithm is described in Section 6.4.2 of your textbook.

Setup

In CodeLite on your virtual machine:

- Create a C++ project named Project6.
- Create a class named DeadlockDetector.

Submitting Your Assignment

Submit the source code (.cpp and .h files) from your project in a zip file. Name the zip file YourLastNameYourFirstNameProject6.zip. Upload the zip file to Canvas before the due date.

Important submission notes for students working in a group:

If you worked with another student, name the zip file using the last names of both students: FirstPersonsLastNameSecondPersonsLastNameProject6.zip. One person should upload the zip files to Canvas. The other person should upload a single document stating who their team member was.

Remember that late assignments are not accepted in this course. See the syllabus for details.

Command Line

Your program must accept a single command-line argument, the name of the input file.

Input File Format

All fields on a line are separated by whitespace.

The first line of the input file contains two integers, the number of processes and the number of resource types. The next line contains the total number of resources of each type (vector E). The next lines contain the data for the current allocation matrix C (first row, followed by the second row, and so on). There will be one line for each process in this matrix. The final set of lines contain the data for the request matrix R. Again, there will be one line for each process.

Sample Input File

```
4 5
2 4 1 4 4
0 1 1 1 2
0 1 0 1 0
0 0 0 0 1
2 1 0 0 0
1 1 0 2 1
0 1 0 2 1
0 2 0 3 1
0 2 1 1 0
```

Assignment

Your program must read from a file the following inputs: the number of processes, the number of resource types, the number of resources of each type in existence (vector E), the current allocation matrix C (first row, followed by the second row, and so on), and the request matrix R (first row, followed by the second row, and so on). When a process is able to finish, your program must print out the process number and the updated available resources (vector A). At the end, your program must indicate whether there is a deadlock in the system or not. In the case of a deadlock, you program must print out the identities of all processes that are deadlocked. To simplify things, we will assume that processes can be identified by their row number in the C and R matrices. The first row contains data for process 0, the second row contains data for process 1, and so on.

Implementation Notes

- You may choose which data structures you want to use.

DeadlockDetector Class

- The constructor must have one parameter, the name of the input file.
- The DeadlockDetector Class must have one public member function named **run**.
- You may add private member variables and private member functions as needed.

Your main method must:

- Check to see if the correct number of command line arguments is given.
- Create an object of type DeadlockDetector.
- Call the DeadlockDetector object's run method to start the detector.

Sample Output for Example Input File

```
Initial Values:
Number of processes: 4
Number of resource types: 5
Existing Resource Vector
2 4 1 4 4
Current Allocation Matrix
0 1 1 1 2
0 1 0 1 0
0 0 0 0 1
2 1 0 0 0
Request Matrix
1 1 0 2 1
0 1 0 2 1
0 2 0 3 1
0 2 1 1 0
Available Resource Vector
0 1 0 2 1
```

```
Process 1 marked
Available Resource Vector
0 2 0 3 1

Process 2 marked
Available Resource Vector
0 2 0 3 2

System is deadlocked
Deadlocked processes: 0 3
```

Grading Criteria (15 points possible)
Note: Your program must compile to receive credit for this assignment!

Points	Criteria
0-1 point	**Main Function:** Is the main function implemented as described in the assignment? Does it accept one command line argument, the name of the input file? Is a “usage” error message printed if the number of command lines arguments is not correct? Does the main function create an instance of the DeadlockDetector class and call the run method?
0-1 point	**DeadlockDetector Class Structure:** Does the program implement a DeadlockDetector class? Are function prototypes and member variables declared in the header (.h) file? Are all member variables private? Are all member functions implemented in the .cpp file of the corresponding class?
0-2 points	**DeadlockDetector Class:** Does the DeadlockDetector class constructor have one parameter as described above? Does it contain one public member function named “run”? Are private member function used to divide up the code into logical segments?
0-8 points	**DeadlockDetector Correctness:** Does the deadlock detection work correctly? Are all vectors and matrices initialized with the correct values? Are processes that are able to finish correctly identified? Is the available resource vector correctly updated? Does the program correctly identify whether the system is deadlocked or not? If the system is deadlocked, are the deadlocked processes correctly identified?
0-2 points	**DeadlockDetector Output:** Are all initial values printed to the console? When a process is able to finish are both the process number and the updated available resources vector displayed? Does the output indicate whether there is a deadlock in the system or not? In the case of a deadlock, are the deadlocked process numbers displayed?
0-1 point	**Style:** Is the code easy to read? Does it follow class style guidelines (See Program Style page in Canvas.)