Group Project #2

CSCE 4600 – Section 002

Course Name: Operating System Design

Spring 2022

**Report**

In order to determine whether the state represented by M is a deadlock, our team decided to implement the graph reduction algorithm. This method consist of two distinct steps:

* select an unblocked process pi
* remove pi, including all the request and allocation edges connected to this process.

These two steps are being repeated until there is no unblocked process remaining in the graph. If the graph is completely reducible (if all processes have removed from the graph), then we assumed there is no deadlock. Otherwise, we assume there is case of deadlock.

When testing the program, it prints the iterations cycling through the graph and if any nodes are removed. Each cycle is separated by a line of exclamation points. If any nodes were unblocked, they will be printed. If none were unblocked in a cycle, no additional nodes will be printed and there is a deadlock.

To test the program run the command:

* **g++ project2.cpp**
* **./a.out inputfile\_name**

Some benefits of the graph reduction algorithm are the following: This method is straightforward and provide and accurate result. If gives a 100% confirmation on the graph status by telling whether there is a deadlock or not. However, this method is not very efficient because every node needs to be removed. Compared to the other method that just look for a node, the graph reduction algorithm takes more time to execute. The execution time depends on the size of the input. The worst case happens when a graph is completely reducible as all the nodes will need to be removed.

* Megan implemented the reading of the input file and formatting the input information and adjacency matrix.
* Dagar implemented the graph reduction algorithm that determines if there is a deadlock or not.
* Suzanne helped with the algorithm and the report.