

CS 3113 Fall 2025 – Project 3
Due Wednesday, November 19th at 11:59 PM

In this project you need to implement the deadlock avoidance algorithm using C++. The Banker's algorithm consists of:

- the safety algorithm
- the resource-request algorithm

Below is the sample input.txt

```
R 3
P 5
Available
3 3 2
Max
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Allocation
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
P1 1 0 2
```

Input explanation:

- R 3 means there are 3 resource types: R0, R1, and R2.
- P 5 means there are 5 processes: P0, P1, P2, P3, and P4.
- Available is a vector with 3, 3, and 2 representing the number of available instances for R0, R1, and R2 respectively.
- Max is a matrix, where each row represents the maximum number of instances of each resource type (R0, R1, R2) needed by the corresponding process.
- Allocation is a matrix, where each row represents the number of instances of each resource type (R0, R1, R2) currently allocated to the corresponding process.
- P1 1 0 2 means P1 is requesting 1 instance of R0, 0 instances of R1, and 2 instances of R2 respectively.

The output should look as follows

```
Before granting the request of P1, the system is in safe state.
Simulating granting P1's request.
New Need
7 4 3
0 2 0
6 0 0
0 1 1
4 3 1
P1's request can be granted. The system will be in safe state.
```

Other Input – Output examples:

Input 2

```
R 3
P 5
Available
3 3 2
Max
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Allocation
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
P4 3 3 0
```

Output 2

```
Before granting the request of P4, the system is in safe state.
Simulating granting P4's request.
New Need
7 4 3
1 2 2
6 0 0
0 1 1
1 0 1
P4's request cannot be granted. The system will be in unsafe state.
```

Input 3

```
R 3
P 5
Available
3 3 2
Max
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Allocation
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
P4 3 3 0
```

Output 3

```
The current system is in unsafe state.
```

TODO

- Write the Banker's algorithm in a file called project3.cpp.
- Assume that the input is read via redirection
- Compile and run project3.cpp. Make sure you keep the input file (e.g input.txt) in the same directory where your program is located.
- Submit project3.cpp to GradeScope.

Linux User

Open terminal and execute the following commands:

```
g++ -std=c++11 project3.cpp -o project3
./project3 < input.txt
```

MacOS User

Open terminal and execute the following commands:

```
clang++ -std=c++11 project3.cpp -o project3
./project3 < input.txt
```

If you do not have clang++ in your MacOS, it means you first need to install XCode by running the following command on the terminal:

```
xcode-select --install
```

Windows User

If you have installed VSCode for the Data Structure class, you can use its terminal to run the same command as in Linux.

If you do not have C, C++ compiler installed in your Windows machine, you can try installing it using one of the following options:

- <https://code.visualstudio.com/docs/cpp/config-mingw>

Rules and Submission Policy

All projects in this course are **individual assignments** and are not to be completed as group work. Collaboration with others or the use of outside third parties to complete this project is strictly prohibited. Submissions must be made through **GradeScope**, where automated grading will be conducted. Additionally, manual grading may be performed to ensure correctness and adherence to requirements.

Several input files will be used to evaluate your program. While a subset of these input files will be provided to you for testing, additional files not shared beforehand will also be used during grading. Your score on this project will depend on producing correct results for all input files, including the undisclosed ones used in GradeScope evaluation.

All programs must be written in **C or C++** and must compile successfully using the **GCC** or **GNU C++ compiler**.

It is your responsibility to ensure that your program adheres to these requirements.

The course syllabus provides more details on the late and submission policy. You should also go through that.