Explanation

The banker algorithm is used in deadlock prevention by the desired resources of each process with the currently available resources. It checks if multiple processes are safe to allocate and, if it is possible, creates a safe sequence based on the currently available resources. After the safe sequence is found, the processes will be allocated in that order. The number of resources needed can be calculated by subtracting the max resources with the number of currently allocated resources.

I decided to use C language again for this assignment, much like what I did for the last assignment with the bounded buffer problem. I used Windows Subsystem for Linux with Ubuntu and Visual Studio Code again. It allows me to test my program without having to dual boot into linux or use a virtual machine. Additionally, I decided to use this assignment as a learning opportunity to use makefiles with my program; letting me easily compile, test, and clean up my program without having to remember and input the command lines every time.

It was a lot quicker for me to get an idea of what I had to do for this assignment compared to the bounded buffer problem. I did not have to learn too much in comparison, it helped more that GeeksForGeeks had a clear solution of the assignment available, which I based a lot of the program around. The makefile took some tinkering, but luckily I found a video from Paul Programming that had everything I was looking for. The biggest issues came from having to read from the input text file and writing it to the arrays.

I created a structure for storing allocated resources, max resources, and needed resources for each process called “customer” . I passed each process's allocation and max arrays into the getDataFromLine() function along with the file variable. Before, I would have the program check each character in the line and try to find a digit. This caused segmentation faults and inconsistencies, so I had to rewrite my function. I used the string.h library’s’ strtok function to omit any parts of the input file that were not a digit. With each string returned, I would use atoi() to convert the string into a digit and place them into the customer arrays.

After each processes’ data was written in and the need array was calculated, I would use the safety algorithm to check if a processes’ needed resources was less than the available resources. In this case, it would mean it is safe to allocate resources with that process, this would be done by adding the allocated resources of the safe process to the available resources. Alongside this, the safe sequence of each process is being put into order and whether a process is safe or not, which is stored in a boolean array. After the loop finishes, the program checks the array containing the booleans for any false flags, which would signify an unsafe process, thus determining the sequence is unsafe. If the sequence is safe, then it will print out the safe order of processes and end the program.

Resources Used

[GeeksForGeeks - Banker's Algorithm]
[GeeksForGeeks - strtok()]
[W3Schools - C Read Files]
[Paul Programming - How To Make a Simple Makefile]

Screenshots

- Makefile being used to create the object files and executable

```
slimuel@The_Scientist:~/vscode-projects/Samuel_Markus-Assignment_2$ make
gcc -c main.c
gcc -c filereader.c
gcc main.o filereader.o -o output
slimuel@The_Scientist:~/vscode-projects/Samuel_Markus-Assignment_2$
```

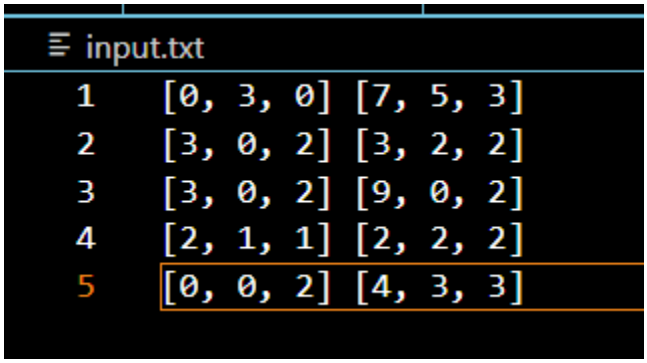
- Example input file and results with available resources set at (3, 3, 2)

input.txt

```
1   [0, 1, 0] [7, 5, 3]
2   [2, 0, 0] [3, 2, 2]
3   [3, 0, 2] [9, 0, 2]
4   [2, 1, 1] [2, 2, 2]
5   [0, 0, 2] [4, 3, 3]
```

```
slimuel@The_Scientist:~/vscode-projects/Samuel_Markus-Assignment_2$ ./output
Following is the SAFE sequence
P1 ->P3 ->P4 ->P0 ->P2
slimuel@The_Scientist:~/vscode-projects/Samuel_Markus-Assignment_2$
```

- Unsafe input file example with available resources set at (2, 1, 0)




```
slimuel@The_Scientist:~/vscode-projects/Samuel_Markus-Assignment_2$ ./output
The following system is not safe.
slimuel@The_Scientist:~/vscode-projects/Samuel_Markus-Assignment_2$
```

- Makefile being used to clean up project folder

```
slimuel@The_Scientist:~/vscode-projects/Samuel_Markus-Assignment_2$ make clean
rm *.o output
slimuel@The_Scientist:~/vscode-projects/Samuel_Markus-Assignment_2$
```