Assignment 4 Report
Name: Miguelangel Tamargo
Panther ID: 5866999

In this assignment, I developed a C program that simulates the Banker's Algorithm to ensure safe resource allocation in a multi-process system. The program allows multiple customers to request and release resources dynamically while the system checks for safe states before granting any requests. The primary objective was to prevent deadlocks by ensuring that the system never enters an unsafe state, adhering to the principles of the Banker's Algorithm.

The design choice centered around creating a flexible and scalable system. I used dynamic memory allocation to handle varying numbers of customers and resource types, allowing the program to adjust based on the input provided at runtime. The key data structures include the available array for the resources currently available, and the maximum, allocation, and need matrices to track each customer's resource usage and requirements. By reading the maximum resource demands from a file, the program can easily adapt to different scenarios without hardcoding values.

One of the main challenges encountered was implementing the safety algorithm efficiently to check for safe states after each resource request. Ensuring that the algorithm correctly simulates future allocations without altering the actual state required careful handling of temporary variables and rollback mechanisms. I addressed this by creating a function is_safe() that performs the necessary checks using copies of the current state, allowing the program to revert to the original state if a request would lead to an unsafe condition.

Another challenge was handling user input and parsing commands accurately. The program needed to accept various commands for requesting and releasing resources while validating the input to prevent errors. I implemented command parsing using strtok() and ensured that the program provides clear feedback and error messages, guiding the user through correct usage.

This assignment was both challenging and pretty enlightening. Working with C at this level really made me dig deep into how resource management and deadlock avoidance work under the hood. Dealing with dynamic memory allocation was a bit tricky, especially making sure there were no memory leaks or corruption. Implementing the Banker's Algorithm forced many challenges, but seeing it function correctly and prevent unsafe states was super rewarding. It gave me a newfound appreciation for how operating systems handle these complex tasks seamlessly and acknowledging just how giant the shoulder we stand are really are.

To conclude, this project not only boosted my programming skills but also my confidence in tackling intricate algorithms in C. It was a tough but enjoyable experience that made me an even more excited for the next assignment and looking to creating similar in other low level learning languages like Rust to build my understanding.