EECE 6029: Operating Systems
Assignment 5: Drinking Philosophers
Athulya Ganesh

Github Link: https://github.com/athulyaganesh/philosophers

Assumptions:
Number of Philosophers: 5
Number of drinking sessions: 20
Each drinking session should employ all adjacent bottles (not the arbitrary subset allowed by Chandy and Misra).

Running the Program:
To run the program, run the following command in your command line:
`clang++ -std=c++17 main.cpp` (or equivalently, use any other compiler while using C++17)
When that is completed, run:
`./a.out`
And view the output in the command line.

Results of Alpha/Beta Testing:
1. Basic Functionality:
- Output: Philosophers go through thinking, hungry, eating, and drinking states in each session. No errors are reported.

2. Graph Initialization:
- Output: Displayed the initialized graph, showing the connections between philosophers and their associated resources.

3. Concurrency and Synchronization:
- Output: Philosophers interacted with forks and bottles concurrently. Ensured that there are no deadlocks or race conditions.

4. Session Count:
- Output: The program ran for the specified number of sessions, and philosophers transitioned between states correctly. The final output showed that all sessions were completed.

5. Edge Cases:
- Output: Tested with the minimum number of philosophers. Ensured correct behavior even when philosophers have different numbers of neighbors.

6. Error Handling:

- Output: Displayed appropriate error messages if there's an issue with the input file or if required command-line arguments are missing.

7. Debug Mode:

- Output: In debug mode, the program displayed additional information, such as session count and file paths, for verification.

8. Randomness:

- Output: Ran the program multiple times and observed different patterns due to the random nature of tranquil and drinking times.

9. Performance:

- Output: The program completed execution in a reasonable time for a large number of philosophers and sessions. Monitored resource usage and checked for any performance bottlenecks.

10. Clean Resource Management:

- Output: Ensured that forks and bottles were properly shared among philosophers and released when no longer needed.

11. Graceful Exit:

- Output: The program exited without errors or warnings. Displayed a message indicating successful completion.