In this homework, we implement Dining Philosopher problem with barrier.

Flow of the program:

- Firstly, walking of philosopher is implemented by sleep them between the 1 and 10 seconds

- Barriers is used because philosophers want to start dining when all philosophers come to table. Firstly, all barriers are released, then when we are available to put plate on the table, we put it with given PutPlate_GUI then we acquire the barrier, whose plate is put on table.

- Then, we make sure that, our mutex is released position (357-363). Because if we do not make sure, we may try to release mutex, that already release and this causes crash

- In here, we make sure that all philosophers reached the table. Then they can start dining

- Then our main loop starts. Firstly, we sleep philosopher between 1 to 10 seconds which, represents that, philosopher is thinking.

- Then, after thinking, philosopher becomes hungry, then we enter the critical region. Firstly, we record the fact that philosopher is hungry, then he/she tries to acquire two forks with test function. This function looks both right and left side of the current philosopher. If both right and left philosopher do not eat and current philosopher is hungry, then current philosopher state is changed as EATING and semaphore, which is related to this philosopher is released. Finally we exit critical region with releasing the mutex and we adjust semaphore so that, if forks were not acquired.

- Then philosopher can eat his/her spaghetti with ForkTake_GUI and Eating_GUI.

- Then, after eating, philosopher should put forks, that he/she used. Therefore, we use StopEating_GUI for represent that the stop eating, then we enter the critical region with mutex.acquire(). Next, we record that, philosopher finished his eating as make state as THINKING. Then we check are right and left of philosopher be able to eat, if so they can start eating. Finally we exit the critical region with releasing the mutex.