**QUESTION 1 README FILE**

**PART A1:**

**Text

Description automatically generated**

Initializing the number of philosophers and defining the left-hand side and right-hand philosopher of the current philosopher.A

**Text

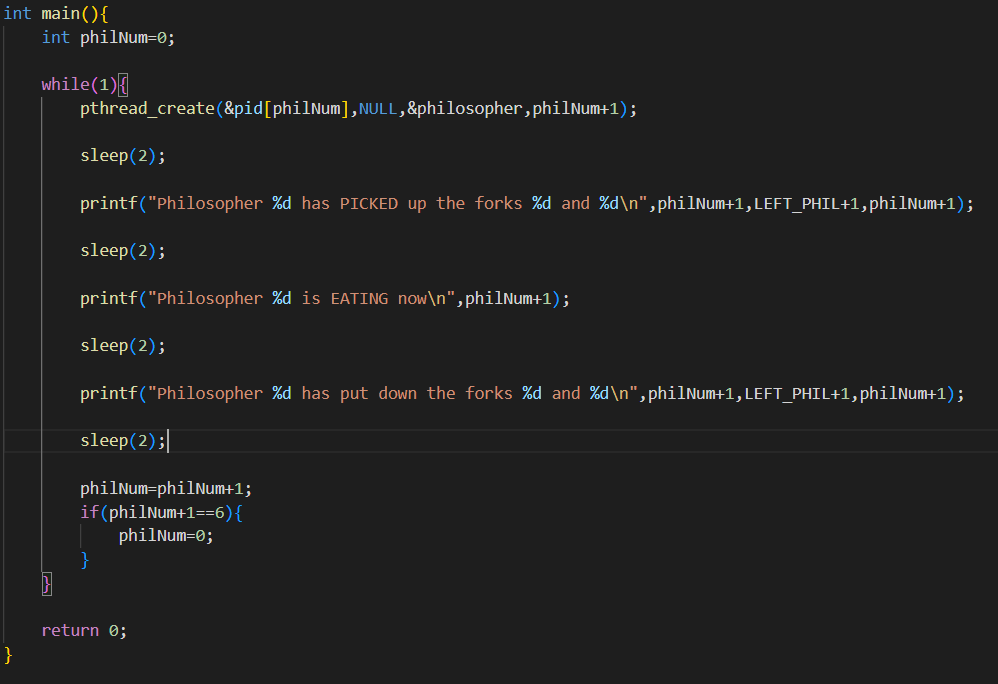
Description automatically generated**

Initializing the philosopher array and thread id array.

**Text

Description automatically generated**

This function prints out the philosopher that is in HUNGRY state. (I have not printed the THINKING state since everyone else other than the current philosopher would be in THINKING state because of strict ordering).



Running an infinite loop, pthread\_create 🡪 creates 5 threads 1 for each of the philosopher.

sleep(2) is used to slow down the output for personal convenience. 1st printf prints the current philosopher who has picked up the forks to eat from the bowl along with the forks that the philosopher has picked up. 2nd printf statement prints the philosopher who is in the EATING state. 3rd printf statement prints the philosopher who has eaten their food and the forks that they’ve placed down.

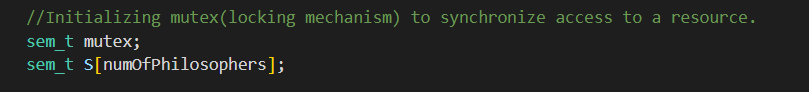
philNum (philosopher number is increased by 1 until it reaches 6 [ if it reaches 6 that means philosopher 5 is the last one to eat], once it reaches 6 it resets back to 0.)

**PART A2**

**Text

Description automatically generated**

Defining each of the states as integers for ease.

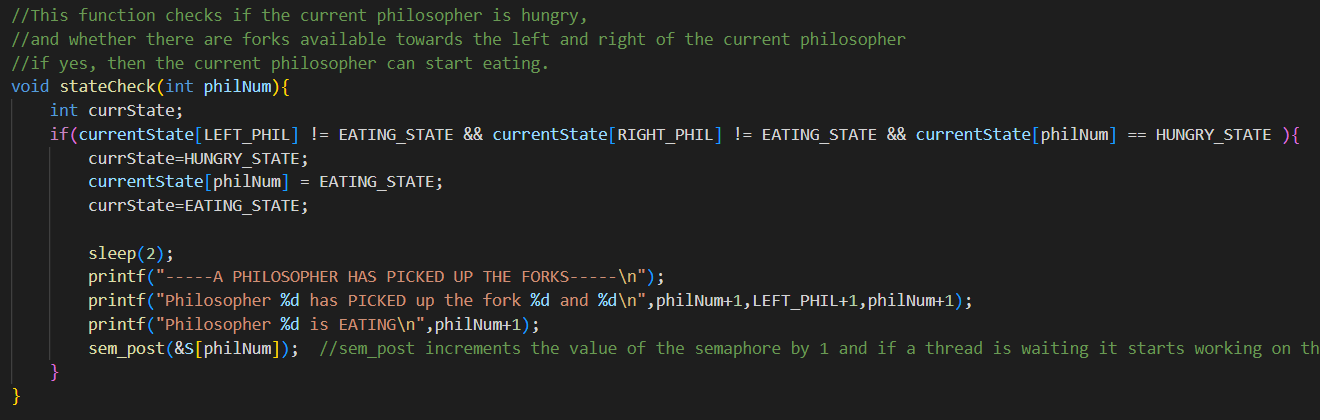


Initializing semaphore.

Text

Description automatically generated

Already explained in the part above.



This function checks if the current philosopher is HUNGRY and whether there are forks available towards his left and right side.

This is a state checking function.

If a philosopher picks up the forks a signal is sent and the value is stored against that philosopher’s index (To avoid deadlock).

Text

Description automatically generated

In this function a HUNGRY philosopher picks up a pair of forks and prints the philosopher who is HUNGRY at the moment. stateCheck function is used to check the current state of the philosopher and the 2 philosophers around them.

sem\_post is used to increment the value of the semaphore.

sem\_wait is used to decrement the value of the state of the philosopher.

**PART B1 and B2** are similar to the above parts because at any point of time there wouldn’t be possibility such that more than 2 philosophers can eat at a particular moment.

**References:**

<https://www.geeksforgeeks.org/dining-philosopher-problem-using-semaphores/>

<https://www.geeksforgeeks.org/mutex-vs-semaphore/>

<https://www.ibm.com/docs/en/i/7.2?topic=ssw_ibm_i_72/apis/ipcsemp.html>