Directory Structure

- **Assignment_5**
    - Q1.c - Program for Q1 ie only 2 forks needed to eat
    - Q2.c - Program for Q2 ie. only one bowl and 1 fork needed to eat
    - Q3.c - Program for Q3 ie. only one bowl and 2 forks needed to eat
    - makefile
    - readme.pdf

Input

No input needs to be given for running the program.

Output

- Q1 will print the current eating status when only two forks are needed to eat and 5 forks and 5 philosophers are present
- Q2 will print the current eating status when only one fork is needed to eat along with a bowl and 5 forks, 4 bowls, and 5 philosophers are present
- Q3 will print the current eating status when 2 forks are needed to eat along with a bowl and 5 forks, 4 bowls, and 5 philosophers are present.

Deadlock Conditions

Q1- There would be a deadlock when the P0 holds F0, P1 holds F1, P2 holds F2, P3 holds F3 and P4 holds F4. In this case, all of them would hold and wait forever resulting in a deadlock. The solution to this problem is given in Q1.c by using a mutex called a waiter.

Q2 - There would be no deadlock in this problem. It is because there are five philosophers and five forks present and 4 bowls. Now since the number of forks is the same as philosophers and each philosopher needs just a single fork all of them could eat at the same time without deadlock. Now since there are 4 bowls 4 philosophers can easily eat simultaneously and then when one of them frees the bowl the other remaining philosopher can eat without any deadlock

Q3- In this case, the deadlock would occur as the forks could create a deadlock similar to the Q1 and one such deadlock would be P0 has F0 and F1, P1 has B0 F2, P2 has B1 F3, P2 has B1 F3, P3 has B2 F4 and P4 has B3. In this case, all of them would wait forever resulting in deadlock.

Expected Output Q1

```
[kern@artixcse231 Assignment_5]$ make
gcc Q1.c -lpthread -o Q1
gcc Q2.c -lpthread -o Q2
gcc Q3.c -lpthread -o Q3
[kern@artixcse231 Assignment_5]$ ./Q1
Philosopher 0 eating with forks 0 and 1
Philosopher 1 eating with forks 1 and 2
Philosopher 3 eating with forks 3 and 4
Philosopher 2 eating with forks 2 and 3
Philosopher 4 eating with forks 4 and 0
Philosopher 0 eating with forks 0 and 1
Philosopher 3 eating with forks 3 and 4
Philosopher 1 eating with forks 1 and 2
Philosopher 2 eating with forks 2 and 3
Philosopher 4 eating with forks 4 and 0
Philosopher 3 eating with forks 3 and 4
Philosopher 0 eating with forks 0 and 1
Philosopher 1 eating with forks 1 and 2
Philosopher 4 eating with forks 4 and 0
Philosopher 2 eating with forks 2 and 3
Philosopher 3 eating with forks 3 and 4
Philosopher 0 eating with forks 0 and 1
Philosopher 4 eating with forks 4 and 0
Philosopher 1 eating with forks 1 and 2
Philosopher 2 eating with forks 2 and 3
Philosopher 0 eating with forks 0 and 1
^C
[kern@artixcse231 Assignment_5]$ _
```

Expected Output Q2

```
[kern@artixcse231 Assignment_5]$ ./Q2
Philosopher 0 eating in Bowl 0 with forks 0
Philosopher 1 eating in Bowl 1 with forks 1
Philosopher 4 eating in Bowl 3 with forks 4
Philosopher 2 eating in Bowl 2 with forks 2
Philosopher 3 eating in Bowl 0 with forks 3
Philosopher 3 eating in Bowl 0 with forks 3
Philosopher 4 eating in Bowl 3 with forks 4
Philosopher 1 eating in Bowl 1 with forks 1
Philosopher 2 eating in Bowl 2 with forks 2
Philosopher 0 eating in Bowl 0 with forks 0
Philosopher 4 eating in Bowl 3 with forks 4
Philosopher 3 eating in Bowl 0 with forks 3
Philosopher 0 eating in Bowl 0 with forks 0
Philosopher 2 eating in Bowl 2 with forks 2
Philosopher 1 eating in Bowl 1 with forks 1
Philosopher 2 eating in Bowl 2 with forks 2
Philosopher 1 eating in Bowl 1 with forks 1
Philosopher 3 eating in Bowl 0 with forks 3
Philosopher 0 eating in Bowl 0 with forks 0
Philosopher 4 eating in Bowl 3 with forks 4
Philosopher 0 eating in Bowl 0 with forks 0
Philosopher 2 eating in Bowl 2 with forks 2
Philosopher 1 eating in Bowl 1 with forks 1
Philosopher 3 eating in Bowl 0 with forks 3
Philosopher 4 eating in Bowl 3 with forks 4
^C
[kern@artixcse231 Assignment_5]$
```

Expected Output Q3

```
[kern@artixcse231 Assignment_5]$ ./Q3
Philosopher 0 eating in Bowl 0 with forks 0 and 1
Philosopher 1 eating in Bowl 0 with forks 1 and 2
Philosopher 2 eating in Bowl 0 with forks 2 and 3
Philosopher 3 eating in Bowl 0 with forks 3 and 4
Philosopher 4 eating in Bowl 0 with forks 4 and 0
Philosopher 0 eating in Bowl 0 with forks 0 and 1
Philosopher 1 eating in Bowl 0 with forks 1 and 2
Philosopher 2 eating in Bowl 0 with forks 2 and 3
Philosopher 3 eating in Bowl 0 with forks 3 and 4
Philosopher 4 eating in Bowl 0 with forks 4 and 0
Philosopher 0 eating in Bowl 0 with forks 0 and 1
Philosopher 1 eating in Bowl 0 with forks 1 and 2
Philosopher 2 eating in Bowl 0 with forks 2 and 3
Philosopher 3 eating in Bowl 0 with forks 3 and 4
Philosopher 4 eating in Bowl 0 with forks 4 and 0
Philosopher 0 eating in Bowl 0 with forks 0 and 1
Philosopher 1 eating in Bowl 0 with forks 1 and 2
Philosopher 2 eating in Bowl 0 with forks 2 and 3
Philosopher 3 eating in Bowl 0 with forks 3 and 4
Philosopher 4 eating in Bowl 0 with forks 4 and 0
Philosopher 0 eating in Bowl 0 with forks 0 and 1
Philosopher 1 eating in Bowl 0 with forks 1 and 2
Philosopher 2 eating in Bowl 0 with forks 2 and 3
Philosopher 3 eating in Bowl 0 with forks 3 and 4
Philosopher 4 eating in Bowl 0 with forks 4 and 0
^C
[kern@artixcse231 Assignment_5]$
```

Data Structures and Synchronisation Primitives used

- **pthread_t array -** This array is used to store the 5 pthreads representing five philosophers that we have created
- **sem_t array -** This array is used to store the semaphores representing five forks and 4 bowls.
- **pthread_mutex_t -** This is the mutex data struct that we are using to create a waiter mutex that can hand over the forks and bowls to the philosophers so that there is no possibility of deadlock.

Description of how the program works

- First, we create five pthreads representing the five philosophers.
- Then we create a mutex which represents the waiter who can handover the forks and bowls to the philosophers so that deadlock can't occur as the waiter will handover either both the forks and bowl or nothing to the philosophers at a time so the condition of deadlock is automatically removed
- Then we create five semaphores representing the forks and 4 semaphores representing the bowls. We use semaphores here so that no two philosophers can access the same forks and bowls at the same time.