BONUS ASSIGNMENT

There are 5 Philosophers seated around a circular table with one fork between each pair of philosophers, The philosopher may eat if he can pick up the two forks adjacent to him. One of the adjacent forks may be picked up by any one of its adjacent Philosophers but not both.
To implement this a design protocol is followed to choose the forks among the dining philosophers so that no deadlock occurs. This is achieved by using semaphores.

Let a thread T[i] represent one of the philosophers,

```
So each philosopher follow this code:
For process T[i]
 while true do
   {  THINK;
      PICKUPFORK(fork[i], fork[i+1 mod 5]);
      EAT;
      PUTDOWNFORK(fork[i], fork[i+1 mod 5])
   }
```

There are three states of the philosopher: THINKING, HUNGRY, and EATING.

Data Structures and Semaphore Related Functions used:

int sem_post(sem_t *sem);

int sem_wait(sem_t *sem);

int sem_init(sem_t *sem, int pshared, unsigned int value);

For Question 1:

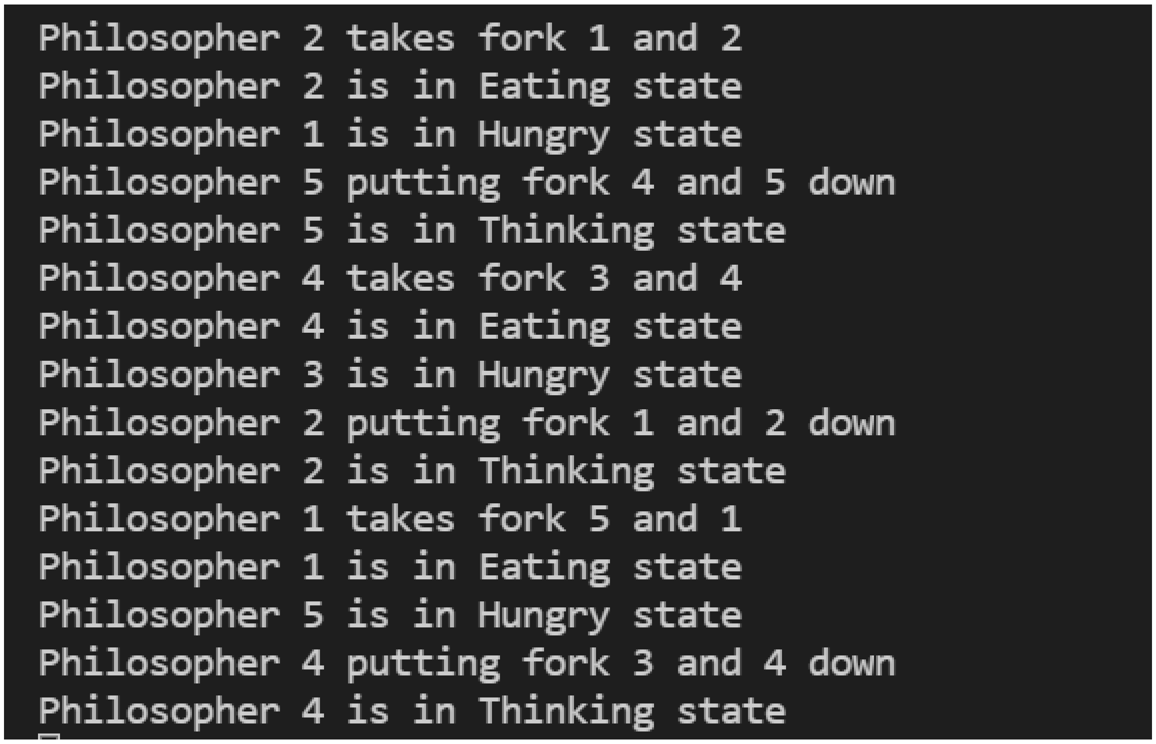
Two sets of semaphores are used,

1) mutex_handle_fork()
2) S[i] where i represents each of the fork from 0 =< i <=5.

mutex_handle_fork() is used such that no two philosophers may access the pickup or putdown of two forks at the same time.
Array S is used to handle the STATE of the philosopher.

Whenever the philosopher is in hungry state it has to wait, till all the adjacent forks are free. So Array S[philosopher_number] is used here.
Philosophers are spawned in thinking state, when a philosopher checks if both the adjacent forks are free to use, if yes then philosopher picks both the forks and its state changes to EATING from hungry state. After putting down both the forks philosopher gets back into the thinking state.

Output:



```
Philosopher 2 takes fork 1 and 2
Philosopher 2 is in Eating state
Philosopher 1 is in Hungry state
Philosopher 5 putting fork 4 and 5 down
Philosopher 5 is in Thinking state
Philosopher 4 takes fork 3 and 4
Philosopher 4 is in Eating state
Philosopher 3 is in Hungry state
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is in Thinking state
Philosopher 1 takes fork 5 and 1
Philosopher 1 is in Eating state
Philosopher 5 is in Hungry state
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is in Thinking state
```

For question 2,

4 Different semaphores are used,

1)sem_t bowl;
2)sem_t mutex_take_fork
3)sem_t mutex_put_fork
4)sem_t S[NO_OF_PHILOSOPHERS];

Bowl is a counting semaphore that is initialized to 4 when 4 different philosophers get hold of one fork and one bowl (so no bowls left), the philosopher even if it has a fork needs to wait till one of the bowls is free.

Array S is used to handle the STATE of the philosopher.

mutex_take_fork() is used such that no two philosophers may access the fork to pick it up at the same time.

mutex_put_fork() is used such that no two philosophers may put down the fork at the same time.

All philosophers when spawned are in a thinking state.
In question two, the philosopher checks if any of the adjacent forks are free, to take them unlike question 1 in which both the adjacent forks are checked if they are free to use.
By taking a fork, the state of the philosopher changes from hungry to eating. After putting down the fork philosopher goes back to the thinking state.

For Question 3:

Semaphores used:

1)sem_t mutex_take_fork;
2)sem_t mutex_put_fork;
3)sem_t S[NO_OF_PHILOSOPHERS];
4)sem_t bowl;

Bowl is a counting semaphore that is initialized to 4 when 4 different philosophers get hold of one fork and one bowl (so no bowls left), the philosopher even if it has a fork needs to wait till one of the bowls is free.

Array S is used to handle the STATE of the philosopher.

mutex_take_fork() is used such that no two philosophers may access the fork to pick it up at the same time.

mutex_put_fork() is used such that no two philosophers may put down the fork at the same time.

Similar to question 1, here the philosopher needs both the adjacent forks, philosophers when in a hungry state wait till it gets access to both the forks.
Once both the forks and the bowl are taken, the philosopher is in an Eating state.
Philosophers after eating drops both the forks so other philosophers could take it, and also free the bowl.
After freeing the bowl and the forks it enters into a thinking state again.