Modified Dining Philosophers Problem

The dining philosophers problem contains five philosophers sitting on a round table can perform only one among two actions – eat and think. For eating, each of them requires two forks, one kept beside each person. Typically, allowing unrestricted access to the forks may result in a deadlock.

(a) Write a program to simulate the philosophers using threads, and the forks using global variables.

I used some well defined process synchronization by using semaphore or mutex and ordering of different functions by using sleep() function .

```
 while(forkstate[f1] != 0){
        printf("Philosper%d is thinking for  fork%d\n",f1,f1);
        sleep(2);}
    forkstate[f1]=1;

    while(forkstate[f2] != 0){
        printf("Philosper%d is thinking for  fork%d\n",f1,f2);
        sleep(2);
        }
    forkstate[f2]=1;

    sleep(2); //eating
```

But the deadlock is well removing by using semaphore this question in a.2 question says that using semaphore :

Defined

```
sem_t chopstick[5];
```

Initialized in main function

```
 for(i=0;i<5;i++)
         sem_init(&chopstick[i],0,1);
```

Use in function by sem_wait and at the end by sem_post the value of it.

```
  printf("Philosopher %d is thinking\n",ph);
         sem_wait(&chopstick[ph]);
         printf("Philosopher %d picks the left chopstick\n",ph);
         sem_wait(&chopstick[(ph+1)%5]);
         printf("Philosopher %d picks the right chopstick\n",ph);
         printf("Philosopher %d begins to eat\n",ph);
         sem_post(&chopstick[(ph+1)%5]);
         sem_post(&chopstick[ph]);
```

Resource : https://dextutor.com/program-on-dining-philosopher-problem/

Now two more bowl is added and processed again by using process synchronization:
Now again :
Defined two bowls with chopstick[5]: semaphore as show

```
sem_t chopstick[5];
sem_t bowl1;
sem_t bowl2;
```

And initiated it in main function:

```
        sem_init (&bowl1,0,1);
                 sem_init (&bowl2,0,1);
         for(i=0;i<5;i++)
         sem_init(&chopstick[i],0,1);
```

Used in function as above used

```
 int philo=*(int *)n;
         printf("Philosopher %d is thinking\n",philo);
         sem_wait(&chopstick[philo]);
         printf("Philosopher %d picks the left chopstick\n",philo);
         sem_wait(&chopstick[(philo+1)%5]);
         printf("Philosopher %d picks the right chopstick\n",philo);
                 sem_wait(&bowl1) || sem_wait(&bowl2);
                 printf("Philosopher %d has bowl to eat\n",philo);
         printf("Philosopher %d begins to eat\n",philo);
         printf("Philosopher %d has finished eating\n",philo);
         sem_post(&chopstick[(philo+1)%5]);
         sem_post(&chopstick[philo]);
        sem_post(&bowl1) || sem_post(&bowl2);
```

Used it well defined way and enjoy :