```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

#define num_philosophers 4

//global array of chopstick locks
sem_t chopsticks[num_philosophers];




//same number of sticks as philosophers
int num_chopsticks = num_philosophers;

//struct to define a philosopher
//We pass this as an argument to the dinner function so we can model each philosopher as a thread
typedef struct {
   char* name;
   char* thought;
   int position;
} philosopher;

void initialize_semaphores() {
   //initialize all the chopstick locks
   int i;

   for(i = 0; i < num_chopsticks; i++) {
      sem_init(&chopsticks[i], 0, 1);
   }



}

void think(char* name, char* thought) {
   printf("%s thinking: %s\n", name, thought);
}

void eat(char* name) {
   printf("%s eating...\n", name);
}
```

```
void *dinner(void *phil) {
   int i;
   philosopher* self = (philosopher *)phil;

   //think, eat, 3 times during dinner
   for(i = 0; i < 3; i++) {
      think(self->name, self->thought);

      //calculate chopsticks to right and left of current philosopher
      int chop1 = self->position;
      int chop2 = (self->position + 1) % num_philosophers;




      //acquire 1st/right chopstick
      sem_wait(&chopsticks[chop1]);
      printf("%s acquires chopstick %d\n", self->name, chop1);

      //add sleep call to ensure context switches & deadlock
      sleep(1);

      //acquire 2nd/left chopstick
      sem_wait(&chopsticks[chop2]);
      printf("%s acquires chopstick %d\n", self->name, chop2);

      //eat!
      eat(self->name);

      //putdown 1st/right chopstick
      sem_post(&chopsticks[chop1]);
      printf("%s puts down chopstick %d\n", self->name, chop1);

      //putdown 2nd/left chopstick
      sem_post(&chopsticks[chop2]);
      printf("%s puts down chopstick %d\n", self->name, chop2);




   }
   printf("%s finished dinner!\n", self->name);
}
```

```
//initializes struct for each philosopher and starts thread for it
void run(pthread_t *threads, char* names[], char* thoughts[]) {
   int i;
   for(i = 0; i < num_philosophers; i++) {
      philosopher *p = malloc(sizeof(philosopher));
      p->position = i;
      p->name = names[i];
      p->thought = thoughts[i];

      pthread_create(&threads[i], NULL, dinner, (void *)p);
   }

   //wait for philosophers to finish dining
   for(i = 0; i < num_philosophers; i++) {
      pthread_join(threads[i], NULL);
   }
}

int main(int argc, char *args[]) {
   //initialize all the semaphores we need
   initialize_semaphores();

   //make our output a bit more readable w/ names and thoughts
   char* names[] = {"Derrida", "Socrates", "Neitzsche", "Kant"};
   char* thoughts[] = {"Hmmmm...", "Values!", "Death!", "Chinese Food!"};

   //model each philosopher as a thread
   pthread_t philosophers[num_philosophers];

   //start threads/philosophers
   run(philosophers, names, thoughts);
   printf("***Dinner is finished***\n");
}
```