```
/* Dining Philosopher's Problem */

#include <stdio.h>
#include <conio.h>

typedef struct {
	int count;
} semaphore;

semaphore fork[5] = { {1}, {1}, {1}, {1}, {1} };
semaphore room = {4};

int i;

void think(int i) {
	printf("\tPhilosopher %d is thinking.\n",i);
}

void eat(int i) {
	printf("\n\tPhilosopher %d is eating.\n",i);
}


int wait(semaphore *s) {

	(*s).count--;

	if((*s).count<0) return 0;
	else return 1;
}

void signal(semaphore *s) { (*s).count++; }

void philosopher(int i, int call) {
	printf("\n");
	think(i);
	if(!wait(&room)) { printf("\n\tThere is no room for Philosopher %d.\n",i); return; }
	if(!wait(&fork[i])) { printf("\tPhilosopher %d cannot lift the left fork.\n",i); return; }
	if(call>0) { philosopher((i+1)%5, call-1); }
	if(!wait(&fork[(i+1)%5])) { printf("\tPhilosopher %d cannot lift the right fork.\n",i);
return; }
	eat(i);
	signal(&fork[(i+1)%5]);
	signal(&fork[i]);
	signal(&room);
}

void main() {

	clrscr();

	philosopher(0, 5);

	getch();

}
```

OUTPUT:

```
	Philosopher 0 is thinking.

	Philosopher 1 is thinking.

	Philosopher 2 is thinking.

	Philosopher 3 is thinking.

	Philosopher 4 is thinking.

	There is no room for Philosopher 4.

	Philosopher 3 is eating.

	Philosopher 2 is eating.

	Philosopher 1 is eating.

	Philosopher 0 is eating.
```