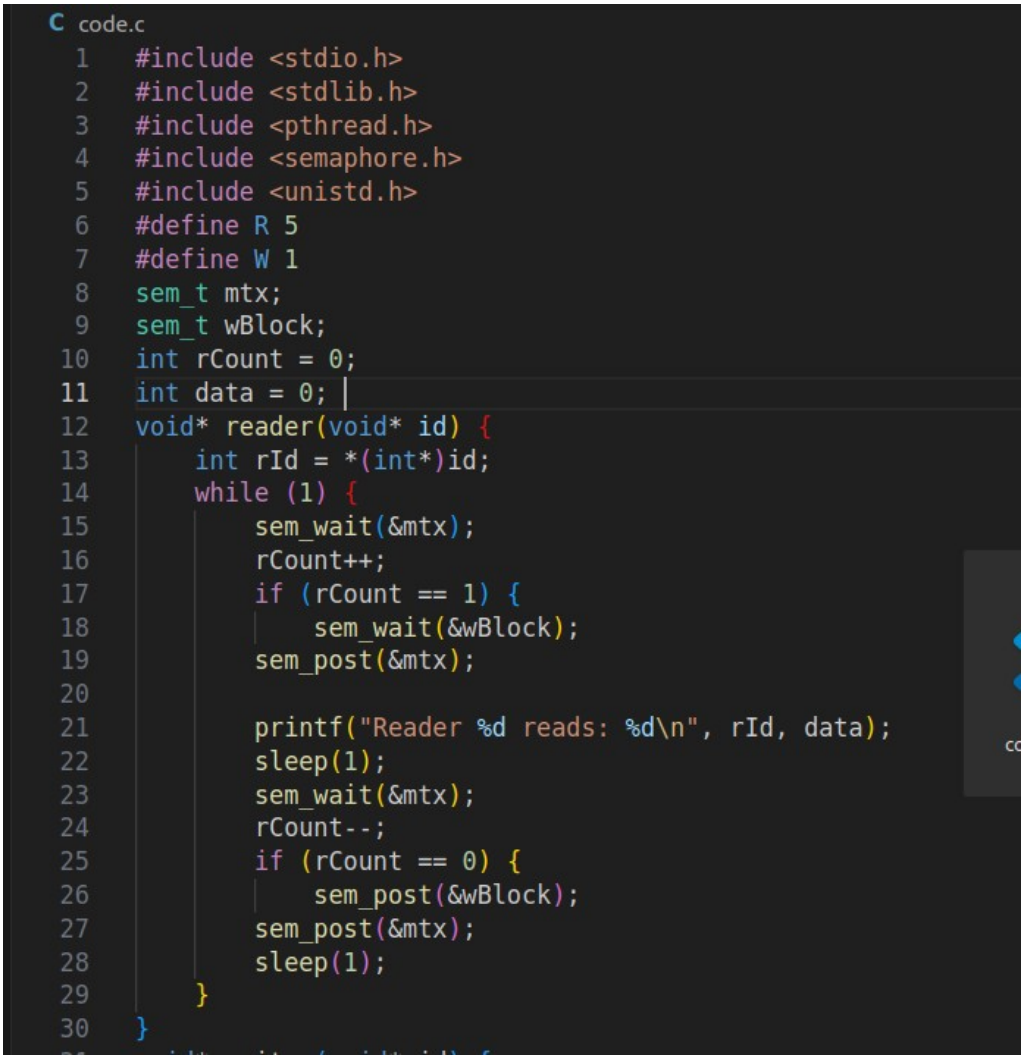
OS LAB 9

23BCE1550
SRIL SHUKLA

1. Write a C program to synchronize Reader and Writer. Considering the reader can only read the shared data whereas a writer can write to the shared data.

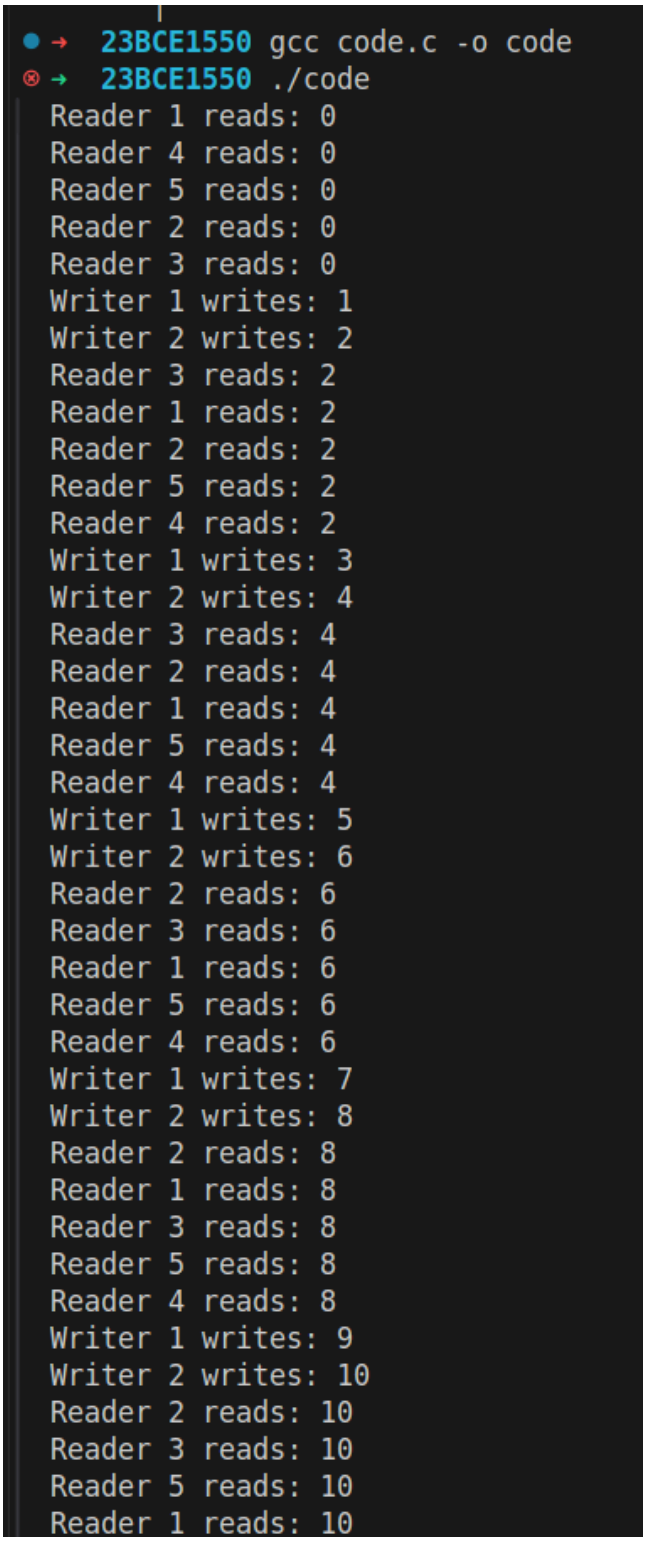
Code:



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define R 5
#define W 1
sem_t mtx;
sem_t wBlock;
int rCount = 0;
int data = 0;
void* reader(void* id) {
    int rId = *(int*)id;
    while (1) {
        sem_wait(&mtx);
        rCount++;
        if (rCount == 1) {
            sem_wait(&wBlock);
        sem_post(&mtx);

        printf("Reader %d reads: %d\n", rId, data);
        sleep(1);
        sem_wait(&mtx);
        rCount--;
        if (rCount == 0) {
            sem_post(&wBlock);
        sem_post(&mtx);
        sleep(1);
    }
}
```

```c
}
void* writer(void* id) {
    int wId = *(int*)id;
    while (1) {
        sem_wait(&wBlock);

        data++;
        printf("Writer %d writes: %d\n", wId, data);
        sleep(2);
        sem_post(&wBlock);
        sleep(2);
    }
}
int main() {
    pthread_t r[R], w[W];
    int rIds[R], wIds[W];
    sem_init(&mtx, 0, 1);
    sem_init(&wBlock, 0, 1);
    for (int i = 0; i < R; i++) {
        rIds[i] = i + 1;
        pthread_create(&r[i], NULL, reader, &rIds[i]);
    }
    for (int i = 0; i < W; i++) {
        wIds[i] = i + 1;
        pthread_create(&w[i], NULL, writer, &wIds[i]);
    }
    for (int i = 0; i < R; i++) {
        pthread_join(r[i], NULL);
    }
    for (int i = 0; i < W; i++) {
        pthread_join(w[i], NULL);
    }
    sem_destroy(&mtx);
    sem_destroy(&wBlock);
    return 0;
}
```

OUTPUT:


23BCE1550 gcc code.c -o code
23BCE1550 ./code
Reader 1 reads: 0
Reader 4 reads: 0
Reader 5 reads: 0
Reader 2 reads: 0
Reader 3 reads: 0
Writer 1 writes: 1
Writer 2 writes: 2
Reader 3 reads: 2
Reader 1 reads: 2
Reader 2 reads: 2
Reader 5 reads: 2
Reader 4 reads: 2
Writer 1 writes: 3
Writer 2 writes: 4
Reader 3 reads: 4
Reader 2 reads: 4
Reader 1 reads: 4
Reader 5 reads: 4
Reader 4 reads: 4
Writer 1 writes: 5
Writer 2 writes: 6
Reader 2 reads: 6
Reader 3 reads: 6
Reader 1 reads: 6
Reader 5 reads: 6
Reader 4 reads: 6
Writer 1 writes: 7
Writer 2 writes: 8
Reader 2 reads: 8
Reader 1 reads: 8
Reader 3 reads: 8
Reader 5 reads: 8
Reader 4 reads: 8
Writer 1 writes: 9
Writer 2 writes: 10
Reader 2 reads: 10
Reader 3 reads: 10
Reader 5 reads: 10
Reader 1 reads: 10

2. Write a C program to synchronize Dinning Philosphor Problem using semaphore.

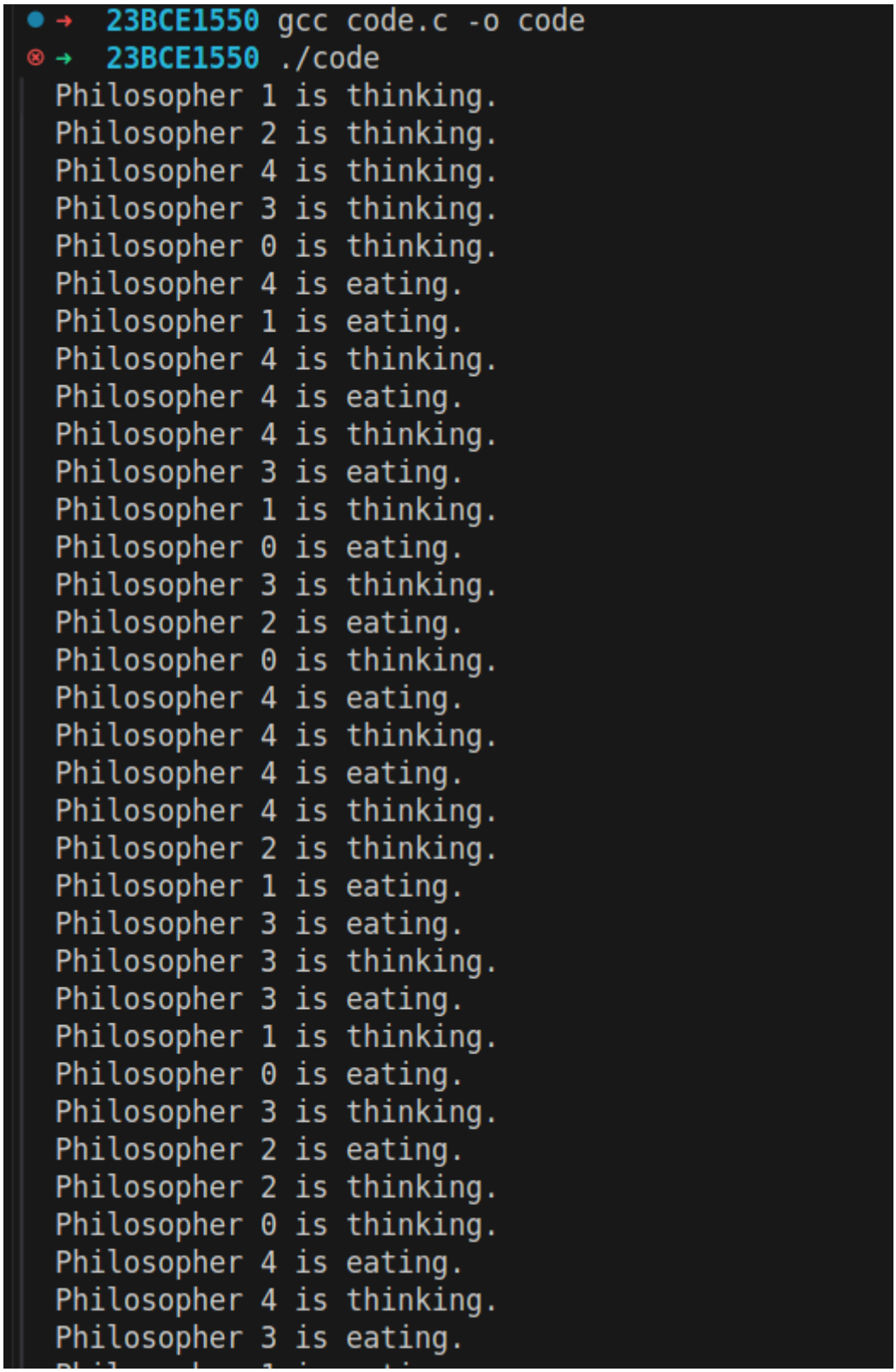
CODE:

C code.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define maxphilo 5
sem_t forks[maxphilo];
pthread_t philosophers[maxphilo];
void* philosopher(void* num) {
    int id = *(int*)num;
    while (1) {
        printf("Philosopher %d is thinking.\n", id);
        sleep(rand() % 3);

        sem_wait(&forks[id]);

        sem_wait(&forks[(id + 1) % maxphilo]);
        printf("Philosopher %d is eating.\n", id);
        sleep(rand() % 3);

        sem_post(&forks[(id + 1) % maxphilo]);

        sem_post(&forks[id]);
    }
}
```

```c
int main() {
    int philosopher_ids[maxphilo];

    for (int i = 0; i < maxphilo; i++) {
        sem_init(&forks[i], 0, 1);
    }

    for (int i = 0; i < maxphilo; i++) {
        philosopher_ids[i] = i;
        pthread_create(&philosophers[i], NULL, philosopher, &philosopher_ids[i]);
    }

    for (int i = 0; i < maxphilo; i++) {
        pthread_join(philosophers[i], NULL);
    }

    for (int i = 0; i < maxphilo; i++) {
        sem_destroy(&forks[i]);
    }
    return 0;
}
```

OUTPUT:


23BCE1550 gcc code.c -o code
23BCE1550 ./code
Philosopher 1 is thinking.
Philosopher 2 is thinking.
Philosopher 4 is thinking.
Philosopher 3 is thinking.
Philosopher 0 is thinking.
Philosopher 4 is eating.
Philosopher 1 is eating.
Philosopher 4 is thinking.
Philosopher 4 is eating.
Philosopher 4 is thinking.
Philosopher 3 is eating.
Philosopher 1 is thinking.
Philosopher 0 is eating.
Philosopher 3 is thinking.
Philosopher 2 is eating.
Philosopher 0 is thinking.
Philosopher 4 is eating.
Philosopher 4 is thinking.
Philosopher 4 is eating.
Philosopher 4 is thinking.
Philosopher 2 is thinking.
Philosopher 1 is eating.
Philosopher 3 is eating.
Philosopher 3 is thinking.
Philosopher 3 is eating.
Philosopher 1 is thinking.
Philosopher 0 is eating.
Philosopher 3 is thinking.
Philosopher 2 is eating.
Philosopher 2 is thinking.
Philosopher 0 is thinking.
Philosopher 4 is eating.
Philosopher 4 is thinking.
Philosopher 3 is eating.

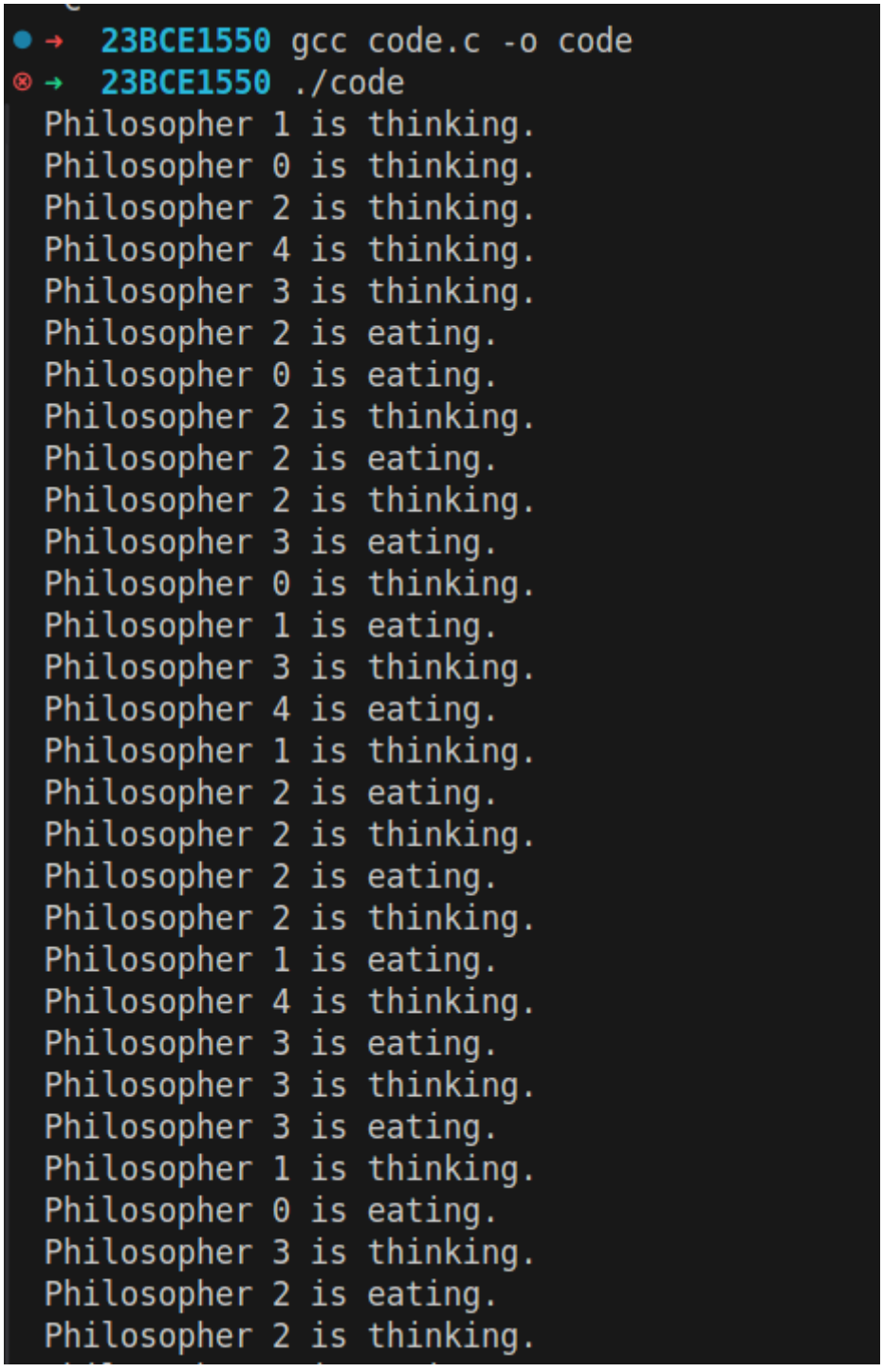
3. Write a C Program to Synchronize Dinning Philospher problem using the monitor

C code.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#define P 5
pthread_mutex_t mtx;
pthread_cond_t cond[P];
int state[P];
void test(int id) {
    if (state[id] == 1
        && state[(id + 4) % P] != 2
        && state[(id + 1) % P] != 2) {
        state[id] = 2;
        pthread_cond_signal(&cond[id]);
    }
}
void take_forks(int id) {
    pthread_mutex_lock(&mtx);
    state[id] = 1;
    test(id);
    while (state[id] != 2) {
        pthread_cond_wait(&cond[id], &mtx);
    }
    pthread_mutex_unlock(&mtx);
}
void put_forks(int id) {
    pthread_mutex_lock(&mtx);
    state[id] = 0;
    test((id + 4) % P);
    test((id + 1) % P);
    pthread_mutex_unlock(&mtx);
}
```

```c
void* philosopher(void* num) {
    int id = *(int*)num;
    while (1) {
        printf("Philosopher %d is thinking.\n", id);
        sleep(rand() % 3);
        take_forks(id);
        printf("Philosopher %d is eating.\n", id);
        sleep(rand() % 3);
        put_forks(id);
    }
}
int main() {
    pthread_t ph[P];
    int pIds[P];
    pthread_mutex_init(&mtx, NULL);
    for (int i = 0; i < P; i++) {
        pthread_cond_init(&cond[i], NULL);
        state[i] = 0;
    }
    for (int i = 0; i < P; i++) {
        pIds[i] = i;
        pthread_create(&ph[i], NULL, philosopher, &pIds[i]);
    }
    for (int i = 0; i < P; i++) {
        pthread_join(ph[i], NULL);
    }
    pthread_mutex_destroy(&mtx);
    for (int i = 0; i < P; i++) {
        pthread_cond_destroy(&cond[i]);
    }
    return 0;
}
```

OUTPUT:


23BCE1550 gcc code.c -o code
23BCE1550 ./code
Philosopher 1 is thinking.
Philosopher 0 is thinking.
Philosopher 2 is thinking.
Philosopher 4 is thinking.
Philosopher 3 is thinking.
Philosopher 2 is eating.
Philosopher 0 is eating.
Philosopher 2 is thinking.
Philosopher 2 is eating.
Philosopher 2 is thinking.
Philosopher 3 is eating.
Philosopher 0 is thinking.
Philosopher 1 is eating.
Philosopher 3 is thinking.
Philosopher 4 is eating.
Philosopher 1 is thinking.
Philosopher 2 is eating.
Philosopher 2 is thinking.
Philosopher 2 is eating.
Philosopher 2 is thinking.
Philosopher 1 is eating.
Philosopher 4 is thinking.
Philosopher 3 is eating.
Philosopher 3 is thinking.
Philosopher 3 is eating.
Philosopher 1 is thinking.
Philosopher 0 is eating.
Philosopher 3 is thinking.
Philosopher 2 is eating.
Philosopher 2 is thinking.