Concurso 4

1.

```c
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

int i = 0;
int sem = 0;

void* produtor(){
    printf("P\n");
    while(1){
        if(i < 10){
            if(sem = 1) sleep(1);
            sem = 1;
            printf("Buffer from\n");
            printf("Prod .%d\n", i);
            i++;
            sem = 0;
        }
        sleep(2);
    }
}

void* consumidor(){
    printf("C\n");
    while(1){
        if(i > 0){
            if(sem = 1) sleep(1);
            sem = 1;
            printf("Buffer out\n");
            printf("Cons .%d\n", i);
            i--;
            sem = 0;
        }
        sleep(4);
    }
}

int main(){
    pthread_t tid[2];
    pthread_create(&tid[0], NULL, produtor, NULL);
    pthread_create(&tid[1], NULL, consumidor, NULL);
    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);
    return 0;
}
```

2. O Jantar dos filosofos é um problema onde 5 filosofos podem estar em 3 estados: HUNGRY, EATING e THINKING. Um filosfo não pode comer comer se um filosfo do seu lado esquerdo ou direito esta a comer.

```c
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>

#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0
#define LEFT (filosofo + 4) % N
#define RIGHT (filosofo + 1) % N

int state[N];
int filosofos[N] = {0, 1, 2, 3, 4};
int priority[N] = {0, 0, 0, 0, 0};

sem_t semaforo;
sem_t sems[N];
pthread_mutex_t mutex;

void try_eat(int filosofo){

    if(state[filosofo] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING){

        if(priority[filosofo] > priority[LEFT] || priority[RIGHT] > priority[LEFT]) return try_eat(LEFT);
        else if(priority[filosofo] > priority[RIGHT])   return try_eat(RIGHT);

        state[filosofo] = EATING;
        priority[filosofo]++;

        sleep(4);
        printf("Philosopher %d takes fork %d and %d\n", filosofo+1, LEFT, filosofo+1);
        printf("Philosopher %d is Eating\n", filosofo+1);

    }

}
```

```c
void is_hungry(int filosofo){


    state[filosofo] = HUNGRY;
    printf("Philosopher %d is Hungry\n", filosofo+1);

    try_eat(filosofo);


    sleep(2);

}

void done_eating(int filosofo){

    state[filosofo] = THINKING;
    printf("Philosopher %d is Thinking\n", filosofo+1);
    try_eat(LEFT);
    try_eat(RIGHT);
    sleep(1);

}
```

```c
void* filosofo_main(void* filosofo){
    while(1){

        int* fil = filosofo;

        sleep(1);

        is_hungry(*fil);

        sleep(1);

        done_eating(*fil);
    }
}

int main(){
    int i;
    pthread_t thread_id[N];

    // initialize the semaphore

    for (i = 0; i < N; i++)

        sem_init(&sems[i], 0, 0);

    for (i = 0; i < N; i++) {

        // create philosopher processes
        pthread_create(&thread_id[i], NULL, filosofo_main, &filosofos[i]);

        printf("Philosopher %d is thinking\n", i + 1);
    }

    for (i = 0; i < N; i++)
        pthread_join(thread_id[i], NULL);

    return 0;
}
```