mutexes program

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

typedef struct town {
    int population;
    pthread_mutex_t * reaper;
} town_t;

typedef struct _move {
    town_t * oldTown;
    town_t * newTown;
} move_t;

void changePopulation(int * population) {
    int currPopulation = *population;
    currPopulation++;

    printf("Population of this town was %i, but is now %i.\n",
           *population,
           currPopulation);

    *population = currPopulation;
}

void incrementPopulation(int * population) {
    int currPopulation = *population;
    currPopulation++;

    printf("Population of this town was %i, but is now %i.\n",
           *population,
           currPopulation);

    *population = currPopulation;
}

void * birth(void * townArg) {
    town_t * town = (town_t *) townArg;

    pthread_mutex_lock(town->reaper);

    incrementPopulation(&town->population);

    pthread_mutex_unlock(town->reaper);

    pthread_exit(NULL);
}

void decrementPopulation(int * population) {
    int currPopulation = *population;
    currPopulation--;

    printf("Population of this town was %i, but is now %i.\n",
           *population,
           currPopulation);

    *population = currPopulation;
}

void * death(void * townArg) {
    town_t * town = (town_t *) townArg;
    pthread_mutex_lock(town->reaper);

    decrementPopulation(&town->population);

    pthread_mutex_unlock(town->reaper);

    pthread_exit(NULL);
}

void * move(void * moveArg) {
    move_t * move = (move_t *) moveArg;

    birth(move->oldTown);
```

```c
    death(move->newTown);

    pthread_exit(NULL);
}

int main() {
    pthread_mutex_t falkirk_reaper;
    pthread_mutex_t stirling_reaper;
    pthread_mutex_t glasgow_reaper;

    pthread_mutex_init(&falkirk_reaper, NULL);
    pthread_mutex_init(&stirling_reaper, NULL);
    pthread_mutex_init(&glasgow_reaper, NULL);

    town_t falkirk = {5, &falkirk_reaper};
    town_t glasgow = {50, &glasgow_reaper};
    town_t stirling = {500, &stirling_reaper};

    move_t ftos = {&falkirk, &stirling};
    move_t gtos = {&glasgow, &stirling};
    move_t stof = {&stirling, &falkirk};

    pthread_t actions[10];
    pthread_create(&actions[0], NULL, birth, &falkirk);
    pthread_create(&actions[1], NULL, death, &falkirk);
    pthread_create(&actions[2], NULL, birth, &falkirk);
    pthread_create(&actions[3], NULL, birth, &stirling);
    pthread_create(&actions[4], NULL, birth, &glasgow);
    pthread_create(&actions[5], NULL, death, &glasgow);
    pthread_create(&actions[6], NULL, death, &glasgow);
    pthread_create(&actions[7], NULL, move, &ftos);
    pthread_create(&actions[8], NULL, move, &gtos);
    pthread_create(&actions[9], NULL, move, &stof);

    int i;
    for (i = 0; i < 10; i++) { // remember to change this when you implement the moves!
        pthread_join(actions[i], NULL);
    }

    printf("\nFalkirk's final population is %d.\nStirling's final population is %d.\nGlasgow's final population is %d.\n\n",
           falkirk.population,
           stirling.population,
           glasgow.population);

    return 0;
}
```