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

#define IMAGE_WIDTH 50
#define IMAGE_HEIGHT 50
#define NUM_THREADS 10

//image[ROW][COLUMN]
color colourImage[IMAGE_WIDTH][IMAGE_HEIGHT];
int greyscaleImage[IMAGE_WIDTH][IMAGE_HEIGHT];

int checkedIndices[IMAGE_WIDTH][IMAGE_HEIGHT];
pthread_mutex_t mutex;

typedef struct color {
    int R;
    int G;
    int B;
} color_t;

void *greyscaleThreadFunction(void *args){

  int i;
  int j;

  //Iterate down each row
  for(i=0;i<IMAGE_WIDTH;i++){
    //Iterate accross each column within current row
    for(j=0;j<IMAGE_HEIGHT;i++){
      //Get the pixel at image[i][j]
      pthread_mutex_lock(&print_state.mutex);

      if(!checkedIndices[i][j]){
        color_t pixel = colourImage[i][j];
        int greyscaleVal = (pixel.R + pixel.G + pixel.B)/3;
        greyscaleImage[i][j] = greyscaleVal;
        checkedIndices[i][j] = 1;
      }

      pthread_mutex_unlock(&print_state.mutex);
    }
  }
  pthread_exit(NULL);
}

int main(){

  //Initialise mutex
  mutex = (pthread_mutex_t)PTHREAD_MUTEX_INITIALIZER;
  pthread_t greyscale_threads[NUM_THREADS];
  int t;

  //Create 10 threads and let them calculate greyscale
  for(t=0;t<NUM_THREADS;t++){

    returnCode = pthread_create(&greyscale_threads[t], NULL,
                            greyscaleThreadFunction, (void *)t);
    if (returnCode) {
                        printf("ERROR return code from pthread_create() : %d\n",returnCode);
                        exit(-1);
                }
  }

  //Wait for all threads to exit
  for(t=0;t<=NUM_THREADS; t++)
        pthread_join(greyscale_threads[t], NULL);

  printf("Successfully exited all threads!\n");
  return(0;)

}
```