```
/*
   Chenxi Guo
   CS6456 Operating Sysytem

   1. Finish all the requirements
   2. If there is any problem, you can contact me through
cx.guo@outlook.com or 6178036588.
   3. Thanks for reading!
*/


#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <pthread.h>

#define NUM_THREADS 1

typedef struct{
    int row;
    int column;
}parameters;

typedef struct{
    int* flag_p;
    char (*wgrid)[9];
    parameters para;
}argument;

int parseInput(char* input, char(*output)[9]){

    int i,j,k=0;

    for(i=0; i<9; i++){
        for(j=0; j<9; j++){
                output[i][j] = input[k];
                k = k + 2;
        }
    }
    return 0;
```

```
}

void *checkSubgrid(void *inputArg){

    int i,j,k,r,c;
    int flag_sg = 0, n = 0, missing_c = 0;
    char newstring[10];
    char missingNum[9];
    argument* arg = (argument*)inputArg;
    r = arg->para.row;
    c = arg->para.column;

    for(i=r;i<r+3;i++){
        for(j=c;j<c+3;j++){
            newstring[n++]=arg->wgrid[i][j];
        }
    }
    newstring[9]='\0';

    for(k=1;k<10;k++){
        char * num = malloc(sizeof(char)*2);
        num[0] = '0'+k;
        num[1] = '\0';
        if(strstr(newstring,num) == NULL){
            missingNum[missing_c++] = num[0];
            *(arg->flag_p) = -1;
            flag_sg = -1;
        }
    }

        if(flag_sg == 0){
            printf("Subgrid       %d...%d,        %d...%d       satisfies        the
requirement.\n",r+1,r+3,c+1,c+3);
        }
        else{
            missingNum[missing_c]='\0';
            printf("Subgrid       %d...%d,        %d...%d        doesn't         have
number %s.\n",r+1,r+3,c+1,c+3,missingNum);
        }

    pthread_exit(0);
}

void *checkRow(void *inputArg){
    int i,j,k;
    int flag_r = 0;
    char newstring[10];
    argument* arg = (argument*)inputArg;
    for(i=0;i<9;i++){
        for(j=0;j<9;j++){
            newstring[j] = arg->wgrid[i][j];
        }
        newstring[9]='\0';

        for(k=1;k<10;k++){
            char * num = malloc(sizeof(char)*2);
            num[0] = '0'+k;
            num[1] = '\0';
            if(strstr(newstring,num) == NULL){
                printf("Row %d doesn't have number %c.\n",i+1,num[0]);
                *(arg->flag_p) = -1;
                flag_r = -1;
            }
        }
    }

    if(flag_r == 0){
        printf("All row satisfies the requirement.\n");
    }

    pthread_exit(0);
}

void *checkColumn(void *inputArg){

    int i,j,k;
    int flag_c = 0;
    char newstring[10];
    argument* arg = (argument*)inputArg;
    for(i=0;i<9;i++){
        for(j=0;j<9;j++){
            newstring[j] = arg->wgrid[j][i];
        }
        newstring[9]='\0';

        for(k=1;k<10;k++){
            char * num = malloc(sizeof(char)*2);
            num[0] = '0'+k;
            num[1] = '\0';
            if(strstr(newstring,num) == NULL){
                printf("Column %d doesn't have number %c.\n",i+1,num[0]);
                *(arg->flag_p) = -1;
                flag_c = -1;
            }
        }
    }

    if(flag_c == 0){
        printf("All row satisfies the requirement.\n");
    }

    pthread_exit(0);
}


int main(){

    char buffer[256];
    char sudoku[9][9];
    char subgrid[3][3];
    int flag = 0;
    char filename[50];
    pthread_attr_t attr;
    pthread_t tid[11];

    //get input numbers
    printf("Please put the input file in the same directory with the csdk.c
file.\nPlease enter the filename:");
    fgets(filename,sizeof(filename),stdin);
    filename[strlen(filename)-1]='\0';
    FILE* pfile = fopen(filename,"r");
    if(pfile != NULL){
        if(fgets(buffer, sizeof(buffer), pfile) == NULL){
            printf("Failed to read file!");
        }
        fclose(pfile);
    }
    else
        printf("Failed to open file!");

    parseInput(buffer, sudoku);
```

```
//check each row
argument* arg[12];
arg[0] = (argument*)malloc(sizeof(argument));
arg[0]->flag_p = &flag;
arg[0]->wgrid = sudoku;
pthread_attr_init(&attr);
pthread_create(&tid[0], &attr, &checkRow, arg[0]);

//check each column
arg[1] = (argument*)malloc(sizeof(argument));
arg[1]->flag_p = &flag;
arg[1]->wgrid = sudoku;
pthread_create(&tid[1], &attr, &checkColumn, arg[1]);

//check each subgrid
int sub =2;
int strt_r, strt_c;
for(strt_r = 0; strt_r<9 && sub<12; strt_r){
    for(strt_c = 0; strt_c < 9 && sub<12; strt_c){

        arg[sub] = (argument*)malloc(sizeof(argument));
        arg[sub]->flag_p = &flag;
        arg[sub]->wgrid = sudoku;
        arg[sub]->para.row = strt_r;
        arg[sub]->para.column = strt_c;
        pthread_create(&tid[2], &attr, &checkSubgrid, arg[sub]);
        strt_c += 3;
        sub++;
    }
    strt_r += 3;
}

//wait for each thread
int thread_c;
for(thread_c=0; thread_c<NUM_THREADS; thread_c++){
    pthread_join(tid[thread_c], NULL);
}

if(flag == 0){
    printf("The solution is correct!\n");
}
else{
    printf("I'm sorry, your solution is incorrect, give another try!\n");
    }

return 0;
}
```