```c
/**
 * Author: Dillon Evans
 * Email: <dillon.e.evans@okstate.edu>
 * Date: March 8th, 2020
 * Program Description: This file creates a socket server that accepts queries from a client.
 * It searches the ID file for the name, then creates two threads to aggregate data
 * efficiently. The threads are then joined and the results are returned to the client.
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <pthread.h>
#include <ctype.h>
#include "defs.h"
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

/*Contains relevant information regarding the file to be read.*/
typedef struct{
    int desiredID;
    char* fileName;
}
fileParam;

/**
 * Compares two strings without case sensitivity.
 * @param str1 The main string
 * @param str2 The string to compare against the main string.
 * @return TRUE if the strings are equal.
 */
bool equalsIgnoreCase(char* str1, char* str2){
    if (strlen(str1) != strlen(str2)){
        return FALSE;
    }
    else{
        for (int i = 0; i < strlen(str1); i++){
            if (tolower(str1[i]) != tolower(str2[i])){
                return FALSE;
            }
        }
    }
    return TRUE;
}

/**
 * Reads from the file specified
 * @param arg A generic argument
 */
void *readFile(void *arg){
    //Implicitly cast the function argument to the struct containing arguments.
    fileParam *p = arg;
    int desiredID = p->desiredID;
    char *fileName = p->fileName;
    FILE* infile = fopen(fileName,"r");
    char buffer[1024] = {0};
    char name[256] = {0};
    bool foundMatch = FALSE;
    int id = 0;
    int count = -1;
    char *returnString = malloc(sizeof(char) * 1024);
    while (fgets(buffer, 1024, infile)){
        if (count >= 0){
            sscanf(buffer, "%d,%[^\r\n]", &id, returnString);
            if (id == desiredID){
                //printf("%s", buffer);
                break;
```

```
            }
        }
        count++;
    }
    fclose(infile);
    return returnString;
}

/*
    Program Entry Point
*/
int main(){

    int server_socket;
    server_socket = socket(AF_INET,SOCK_STREAM, 0);
    struct sockaddr_in server_address;
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(9002);
    server_address.sin_addr.s_addr = INADDR_ANY;
    bind(server_socket, (struct sockaddr*)&server_address, sizeof(server_address));

    listen(server_socket, 1);

    int client_socket;
    client_socket = accept(server_socket, NULL, NULL);

    //Keep the server active until the user terminates the program.
    while(1){

        char query[256] = {0};
        recv(client_socket, query, sizeof(query), 0);
        printf("%s\n", query);
        FILE* infile = fopen("ID_name.txt","r");
        char buffer[1024] = {0};
        char name[256] = {0};
        bool foundMatch = FALSE;
        int id = 0;
        int count = -1;

        //Read the file and scan it
        while (fgets(buffer, 1024, infile)){
            if (count >= 0){
                sscanf(buffer, "%d,%[^\r\n]", &id, name);
                if (equalsIgnoreCase(query, name)==TRUE){
                    printf("MATCH FOUND\n");
                    foundMatch = TRUE;
                    break;
                }
            }
            count++;
        }
        fclose(infile);

        if(!foundMatch){
            printf("The Employee %s Was Not Found.\n", query);
            send(client_socket, INVALID_QUERY, sizeof(INVALID_QUERY), 0);
        }else{
            pthread_t Salaries, Satisfaction;
            fileParam t1 = {id, "Salaries.txt"}, t2 = {id, "SatisfactionLevel.txt"};
            void *result1, *result2;
            //Create two threads
            pthread_create(&Salaries, NULL, readFile, &t1);
            pthread_create(&Satisfaction, NULL, readFile, &t2);

            //Join the two threads and get the results.
            pthread_join(Salaries, &result1);
            pthread_join(Satisfaction, &result2);

            char *str1 = result1, *str2 = result2;
```

```
138                 printf("%s\n", str1);
139                 printf("%s\n", str2);
140                 char returnString[1024] = {0};
141                 sprintf(returnString, "%d,%s,", id, name);
142                 //Join the strings returned from the two threads
143                 strcat(returnString, str1);
144                 strcat(returnString,",");
145                 strcat(returnString, str2);
146 
147                 free(str1);
148                 free(str2);
149                 printf("%s", returnString);
150                 printf("Return String: %s\n", returnString);
151                 //Send the message back to the client.
152                 send(client_socket, returnString, sizeof(returnString), 0);
153             }
154         }
155     }
156 
```