```
/*
         * Author Name: Maksim Tybar
         * Email: mtybar@okstate.edu
         * Date: 03/28/2021
         * Program Description: This program implements the assistant role of the mini
        group project (receiving part)
                                Receiving part send name request to the server and gets
        back the appropriate
                                information and then stores it in the history.txt file.
        */

        #include <stdio.h>
        #include <stdbool.h>
        #include <sys/socket.h>
        #include <arpa/inet.h>
        #include <unistd.h>
        #include <string.h>
        #include <stdlib.h>
        #include <pthread.h>

        #define PORT 8080
        #define BUFFER_SIZE 512

        int lineCount = 0; // To count amount of lines until they reach 10
        int count = 0; // To count each line in the history.txt file to copy it to other
        .txt file
        int lineToRemove = 0; // To count which line should be overwritten in the
        history.txt file

        void *assistantReceive() { // This can be changed for int main() function and
                                   // return NULL at the bottom could be changed for return
        0
            int sock = 0, valread;
            struct sockaddr_in serv_addr;
            char buffer[BUFFER_SIZE];

            char empName[100];
            char empID[100];
            char jobTitle[100];
            char pay[100];
            char overtimePay[100];
            char benefitPay[100];
            char statusReceived[100];
char satisfactionLevel[5];
char numberProject[5];
char monthlyHours[5];
char yearsAtCompany[5];
char workAccident[5];
char promotionGiveYears[5];

if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
    printf("\n Socket creation error \n");
    return -1;
}

serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary form
if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <=0 ) {
    printf("\nInvalid address/ Address not supported \n");
    return -1;
}

if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    printf("\nConnection Failed \n");
    return -1;
}


/*

Get info from the server and write it to the history file below.
*/

/*
    Reading each parameter from the server.
*/

read(sock, empName, 100);
printf("Job title recieved from server: %s\n", empName);

read(sock, empID, 100);
printf("Job title recieved from server: %s\n", empID);
```

```
read(sock, jobTitle, 100);
printf("Job title recieved from server: %s\n", jobTitle);

read(sock, pay, 100);
printf("Pay recieved from server: %s\n", pay);

read(sock, overtimePay, 100);
printf("Overtime pay recieved from server: %s\n", overtimePay);

read(sock, benefitPay, 100);
printf("Benefit pay recieved from server: %s\n", benefitPay);

read(sock, statusReceived, 100);
printf("Status recieved from server: %s\n", statusReceived);

read(sock, satisfactionLevel, 100);
printf("Satisfaction level received from server: %s\n", satisfactionLevel);

read(sock, numberProject, 100);
printf("Number project received from server: %s\n", numberProject);

read(sock, monthlyHours, 100);
printf("Monthly numbers received from server: %s\n", monthlyHours);

read(sock, yearsAtCompany, 100);
printf("Years at company received from server: %s\n", yearsAtCompany);

read(sock, workAccident, 100);
printf("Work accident received from server: %s", workAccident);

read(sock, promotionGiveYears, 100);
printf("Promotion give years received from server: %s", promotionGiveYears);

lineToRemove = (lineToRemove + 1) % 11;
if (lineToRemove == 0) {
    lineToRemove ++;
}

FILE *fp;
FILE *temp;

/*
    If there is 10 entrees in the history file,
    copy everything to the new history file and rewrite
    the appropriate line.
*/

if (lineCount == 10) {
    fp = fopen("history.txt", "r");
    temp = fopen("history_temp.txt", "w");

    while ((fgets(buffer, BUFFER_SIZE, fp)) != NULL) {
        count = (count + 1) % 11;
        if (count == 0) {
            count ++;
        }

        if (lineToRemove == count) {
            fprintf(temp, "%s,", empName);
            fprintf(temp, "%s,", empID);
            fprintf(temp, "%s,", jobTitle);
            fprintf(temp, "%s,", pay);
            fprintf(temp, "%s,", overtimePay);
            fprintf(temp, "%s,", benefitPay);
            fprintf(temp, "%s,", statusReceived);
            fprintf(temp, "%s,", satisfactionLevel);
            fprintf(temp, "%s,", numberProject);
            fprintf(temp, "%s,", monthlyHours);
            fprintf(temp, "%s,", yearsAtCompany);
            fprintf(temp, "%s,", workAccident);
            fprintf(temp, "%s,", promotionGiveYears);
            fprintf(temp, "\n");
        } else {
            fprintf(temp, "%s", buffer);
        }
    }

    fclose(fp);
    fclose(temp);

    remove("history.txt");
    rename("history_temp.txt", "history.txt");

} else {
    lineCount ++;
    fp = fopen("history.txt", "a");
```

```
        if (!fp) {
            perror("Couldn't open history.txt file in the Assistant thread.\n");
            return -1;
        }
        fprintf(fp, "%s,", empName);
        fprintf(fp, "%s,", empID);
        fprintf(fp, "%s,", jobTitle);
        fprintf(fp, "%s,", pay);
        fprintf(fp, "%s,", overtimePay);
        fprintf(fp, "%s,", benefitPay);
        fprintf(fp, "%s,", statusReceived);
        fprintf(fp, "%s,", satisfactionLevel);
        fprintf(fp, "%s,", numberProject);
        fprintf(fp, "%s,", monthlyHours);
        fprintf(fp, "%s,", yearsAtCompany);
        fprintf(fp, "%s,", workAccident);
        fprintf(fp, "%s,", promotionGiveYears);
        fprintf(fp, "\n");

        fclose(fp);
    }

    // int status = system("open -a Terminal \"`pwd`\""); // TO OPEN NEW TERMINAL
    // printf("New terminal is opened.\n");


    return NULL;
}
```