```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include "structure.h"
#define PORT 8080

/*
Assistant.c is the first half of the assistant
This process connects to the server and then starts a loop that reads in employee

info from the fifo pipe, checks the history file for the employee, and sends the
information to the server if the search returns false

Written by Kayla Walkup
*/

//function declaration
int fileSearch();

struct User *userPtr, user;

int main(int argc, char const *argv[])
{

    printf("Assistant Started! \n");

//====== Socket Setup ========================================================
    //initializing socket vars
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    char buffer[1024] = {0};

    //create socket, print error and exit if fail
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("socket");
        return -1;
    }
```

```
    //setting addresses
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    // Converting IPv4 and IPv6 addresses to binary, exit if fail
    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0)
    {
        perror("inet_pton");
        return -1;
    }

//=============================================================================
===




//=====Pipe setup==============================================================
=

    //struct pointer to store employee info
    userPtr = &user;

    //pipe file descriptor
    int fd1;

    // FIFO file path
    char *myfifo = "myfifo.txt";

    //chars to store read data in
    char str1[100];
    char str2[100];
    char str3[100];



    //open pipe for read
    fd1 = open(myfifo, O_RDONLY);

//=============================================================================
===

   //print error if connection fails
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
```

```
        perror("connect");
    }

    //token to split up string from pipe
    char *token;

    //loop to read from pipe, check history file for employee, and send employee
to server if not there
    while(1)
    {
        //reading from pipe if pipe is not empty
        if(read(fd1, str1, 100) > 0)
        {
            //feedback on if read was successful or not
            perror("read");

            //splitting up info from pipe and entering it into the struct
            token = strtok(str1, ",");
            strcpy(userPtr->employeeName, token);
            printf("Name: %s\n", userPtr->employeeName);

            token = strtok(NULL, ",");
            strcpy(userPtr->jobTitle, token);
            printf("Job Title: %s\n", userPtr->jobTitle);

            token = strtok(NULL, ",");
            strcpy(userPtr->status, token);
            printf("Status: %s\n", userPtr->status);
            //-------------------------------------------------------------



            //search file, if employee not in file, send to server
            if(fileSearch() == 1)
            {
                //sending employee information to server from struct
                send(sock, userPtr->employeeName, 100, 0);
                send(sock, userPtr->jobTitle, 100, 0);
                send(sock, userPtr->status, 100, 0);

                //print statement to test the send worked
                printf("Employee sent!\n");
            }
```

```
        }
    }


//=============================================================================
=======


    return 0;
}


//fuction to search history file for employee
int fileSearch()
{
    //opening file------------------
    FILE *filep;
    char *filename = "history.txt";

    filep = fopen(filename, "a+");
    if (filep == NULL)
    {
        perror("Error: ");
        exit(0);
    }
    //-------------------------------

    //return false (1) if file is empty
    int size;
    fseek (filep, 0, SEEK_END);
    size = ftell(filep);
    if (0 == size) {
        //printf("file is empty\n");
        return 1;
    }
    //---------------------------------



    //search the file for employee info----------------------------------------
    char line[256];
    while (fgets(line, sizeof(line), filep))
    {

       if(strstr(line, userPtr->employeeName))
```

```
        {
            if(strstr(line, userPtr->jobTitle) && strstr(line, userPtr->status))
            {
                //Emplyee in history file, can print
                printf("%s\n", line);
                return 0;
            }

        }
        else
        {
            return 1;
        }

        return 1;
    }
    //-------------------------------------------------------------------------
-----

    //close file
    fclose(filep);

    return 1;
}
```