```c
// Author: Michael Royster
// Group D
// Email: micaher@okstate.edu

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <stdbool.h>
#include "Assistant.h"
#include "History.h"
#include "Query.h"
#include "Record.h"

#define QUERY_MAX sizeof(Query)
#define RECORD_MAX sizeof(Record)
#define PORT 8000
#define SA struct sockaddr

void query_server(int sockfd, Query send_query, Record *record);

int Assistant(){
    printf("Client started..\n");
    // Forking for Output
    if(fork() == 0){
        execlp("gnome-terminal", "gnome-terminal", "--", "./output", NULL);
    }
    else{
    // Create socket
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1){
        printf("Socket creation failed..\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");

    bzero(&servaddr, sizeof(servaddr));

    // assign IP and Port
    servaddr.sin_family         = AF_INET;
    servaddr.sin_addr.s_addr    = inet_addr("10.203.72.5");
    servaddr.sin_port           = htons(PORT);

    // Connect to server
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0){
        printf("Connection with server failed...\n");
    }
    else
        printf("Connected to the server..\n");

    // ============= SOCKET HAS CONNECTED TO SERVER - Now send request ===========

    int fd, fd2;
    Query query;
    Record record;
```

```c
    Record *ptr_record = &record;

    // Open manager pipe to read
    char manager_pipe[] = "./ManagerPipe";
    mkfifo(manager_pipe, 0666);
    if ((fd = open(manager_pipe, O_RDONLY)) < 0)
        perror("Assistant: Error opening...");

    // Open assistant pipe for writing
    char assistant_pipe[] = "./AssistantPipe";
    mkfifo(assistant_pipe, 0666);
    if ((fd2 = open(assistant_pipe, O_WRONLY | O_CREAT)) < 0)
        perror("Assistant: assistant_pipe opening...");

    while(1){
        // Read from Manager
        if(read(fd, &query, sizeof(Query)) < 0)
            printf("Assistant: Error reading...\n");

        if (strcmp(query.employee_name, "exit") == 0 && strcmp(query.job_title, "exit") == 0 && strcmp(
query.status, "ex") ==0){
            Record exit_record;
            strcpy(exit_record.employee_name, "exit");

            // tell output terminal to end
            if (write (fd2, &exit_record, sizeof(Record)) < 0)
                printf("Assistant: error terminating output terminal\n");

            // tell server to end
            send(sockfd, &query, sizeof(Query), 0);

            break;
        }

        //Check History File
        if(isInHistory(query)){
            getFromHistory(query,ptr_record);
        }
        else{
            // Send request to server after checking history file if necessary
            query_server(sockfd, query, ptr_record);
            writeToHistory(ptr_record);
        }

        // Write to Output terminal
        if (write(fd2, &record, sizeof(Record)) < 0)
            perror("Assistant: error writing...");

    }

    // Closing pipes and server
    close(fd);
    close(fd2);
    close(sockfd);
    wait(NULL);
    printf("Assistant closed.\n");
    }
    return 0;
}

void query_server(int sockfd, Query send_query, Record *record){
    Record temp;
    int val;

    send(sockfd, &send_query, sizeof(Query), 0);

```

```
132     val = recv(sockfd, &temp, sizeof(Record), 0);
133     if(val == -1){
134         printf("Error recieving data..\n");
135     }
136 
137     *record = temp;
138 
```