Academic Year	2024		
Semester	☐ Fall	☒ Winter	☐ Summer
Course Code - Name	CSCI 3310 – System Programming		
Instructor	Dr. Razi Iqbal		
Assessment	Assignment 3		
Deadline	April 5, 2024		

Instructions:

In order to obtain maximum marks in this assignment, please ensure the followings:

- Please submit your C files (.c) for this assignment.
- This assignment has a weightage of **10%** marks of the course.
- This is **NOT** a group assignment so **students having similar assignments will get a 0.**
- The assignment deadline is **midnight April 5, 2024**. Submissions after the deadline will not be accepted.

Question

Canada is widely recognized as a diverse and inclusive country that embraces people from all backgrounds, cultures, and beliefs. Canada's commitment to diversity and inclusivity is also reflected in its immigration policies, which provide opportunities for people from around the world to settle in Canada and become part of its vibrant and dynamic communities. Canada provides social welfare programs that support those in need, including seniors, people with disabilities, and families with children.

In this assignment, we'll be creating a server that provides information about various benefits Canada as a country provides to its citizens. You are provided with a file (Benefits_Canada.txt) that contains information about various benefits. This is a ";" separated file with each entry described below:
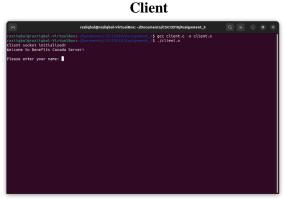
Serial	;	Code	;	Name	;	Details	;	URL	;

Your task in this assignment is to write C programs for both Server and Client using Socket Programming in Linux. Use TCP for communication. A user from the Client computer would request for information on a specific benefit program and Server is expected to send back the details of those benefits to the Client computer. Please complete each step with all of its requirements below to get maximum marks.

Step 1: Establishing connection

In this step you have to establish a connection between the Server and the Client. You are expected to perform all the required pre-requisite, e.g., creating socket, populating server details, listening and accepting connection at the Server side and connecting to the Server from the Client side etc. to establish this connection successfully. Make sure you create a multi-threaded Server which can entertain requests from multiple Clients at the same time. Once the connection is established, Server should send a "Welcome to Benefits Canada Server!" message to the Client. The Client shows this message on its console and should prompt user to enter their name. Below are the expected screenshots of both Server and Client for Step 1.

<table>
<tr><th>Server</th><th>Client</th></tr>
<tr><td></td><td></td></tr>
</table>

Step 2: Send Name and Get Menu

In this step, user inputs their name at the Client side and Client sends this name to the Server. The Server receives the name and sends back a greeting with name and the main menu to the Client. The Client receives the greeting and the main menu and shows that on Client's console. User is also prompted to select one of the menu options from the menu provided. Below are the expected screenshots of both Server and Client for Step 2. For this step, you are most welcome to hardcode a multiline string that shows the menu with their appropriate numbers at the Server side and send it to the Client. Please refer to the screenshot below for how a menu should look like.

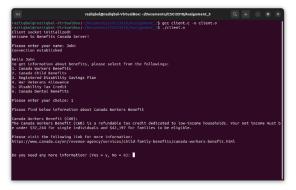| **Server**	**Client**

Step 3: Send Choice and Get Response

In this step, user selects one of the options on the Client side and presses Enter. This choice of options, e.g., 1, 2, etc. is sent to the Server from the client Side. The Server receives this information and checks this Serial in the text file (Benefits_Canada.txt). Server after finding this information from the text file, compiles this information in the form of a char array and sends this information back to the Client. The Client after receiving this information, parses the char array to show information in a presentable manner as shown in the screenshot below. Please note that you have to follow the formation of the information presented at the Client side and your output must match the one shown in the screenshots below. Also, please note Client after showing the information does not exit and asks the user if they want to get any more information from the Server. The Server also shows a message on its console saying "Requested Information Sent":
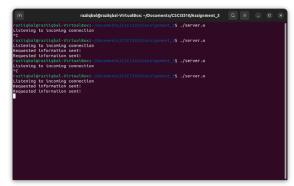
Server	**Client**

Step 4: Requesting more information from the Client

On the Client side, if user presses 'y' as a choice to enter more information (please refer to the screenshot of Step 3), the Client should fetch the greeting with name and menu from the Server again and on user choice of selection of the menu item, should fetch details for that particular program from the Server again. This process should continue till user presses 'n' as a choice if asked for more information as shown in the screenshot of Step 3. If user presses 'n', the Client program should exit, however, Server should keep running to wait for more clients to connect.

Server	Client

Step 5: Logging at Server side

One last step to follow is to make sure that all the tasks happening at the Server side are logged appropriately in a log file. Make sure to tag each message with the date and time of that particular message. Also, log file should contain the code of each benefit requested from the server. Below is the screenshot of the expected output of the log file at the Server side.

Please keep in mind the followings:
- Communication should be TCP
- Server must be a multi-threaded Server
- Benefits_Canada.txt file is available at the Server side and not at the Client side
- Menu must be sent from the Server side
- Information of each benefit must be read from the text file at the Server side and then sent to the Client side
- Client must be able to format the information of the benefits received from the Client in exactly the same manner as shown in the screenshot in Step 3.
- Client program should not end till user selects 'n' as a choice of whether to get more information or not?
- Server should maintain a log in the form of a text file to keep record of what information is requested at what instance from the Client side.
- Server should keep running even if Client program has exits.