Files: ppServer.c, pqClient.c

Pseudocode:

Terminal 1:

-- Start ppServer.c

Create server to listen on localhost (127.0.0.1) port 8080 using code from pages 214-215 in chapter 5 of the textbook (TCP server).

(port 8080 is a commonly used alternative to port 80 that is above the service port range)

Get the file name from the user and open the file

Wait for client connections…

When a client connects wait for an incoming option from the menu displayed on the client-side.

```
(option 1){
    Create a linked list of all Pokemon of an inputted type 1 using recycled code from assignment 3
    Find and send length of linked list
    currNode = head of list
    For (int i from 0 to length){
        Send currnode->data string to client
        Receive ok message from client
        currNode = currNode->next
    }
    Free linked list memory
}
(option 2){
    Send an ok message to the client and handle everything client-side
}
(option 3){
    Disconnect client using code from server.c in the textbook
}
```

```
(hidden option 4 for debugging/admin purposes){
    Disconnect client using code from server.c in the textbook
    Free all heap memory
    Terminate server using code from server.c in the textbook
}
End ppServer.c --
```

Terminal 2:

-- Start pqClient.c

Create a client to connect on localhost (127.0.0.1) port 8080 to ppServer.c using code from pages 216-217 in chapter 5 of the textbook (TCP server).

```
Display a menu with 3 choices
Do{
    (option1){
        Revieve length of linked list
        For(int i from 0 to length){
            Revieve currnode->data string from server
            Add this to a linked list using code from assignment 3
            Send ok message to server
        }
    }
    (option 2){
        Save pokemon data to a file using recycled code from assignment 3
    }
}while(!choice3 && !choice4);
Send choice 3 or 4 to server
Free all heap memory
Terminate client using code from client.c
End pqClient.c –
```

The project will contain 2 c files (ppServer.c and pqClent.c) and 2 header files (ppServer.h and pqClient.h) so that different programmers could work on the client and server portions in parallel. A custom Makefile will also be provided.

All functional and non-functional requirements are described in the comments and pseudocode.

A TCP server will be used rather than UDP for a connection-oriented process with stronger packet loss prevention while rapidly sending large amounts of data through for loops. TCP was also largely advised over UDP in the assignment 4 discussion section of the discord server.

C language features used include:

<stdio.h> for input and output to files and the console

<stdlib.h> for allocating and deallocating heap memory

<string.h> for string manipulation functions like strtok() and strcmp()

<unistd.h> for closing sockets

<pthread.h> for multithreading

<sys/socket.h> <netinet/in.h> <arpa/inet.h>

For TCP server operations such as bind() listen() send() accept() recv()…