Design Doc

Pseudo Code

Server Side

1. Create Server Socket → utilize UDP server model
    a. If server Socket doesn't work, exit program and show error
2. Bind the Socket to the Address
3. Ask user for file name
    a. If file doesn't exist show error and ask again until they get it right
4. While True
    a. Select a socket request
    b. Receive the buffer from the client (the type1)
    c. Search though the entire file
        i. Skip the Header line
        ii. For each line
            1. Strip each line by "," and create a token
            2. If the 3rd token matches with the Buffer
                a. Send the line to the Client
    d. If the buffer is equal to "stop"
        i. Break loop and shut down server

Client Side

1. Create Client Socket → utilize UDP server model
    a. If client socket doesn't work, exit program and show error
2. Bind and set up the Address
3. Create a Linked List structure for file names and Pokémon data
4. Get user input
    a. Make sure its 1,2 or 3 other wise ask again until user get it right
5. If user input is 1
    a. Ask user for Type 1
    b. Send Type 1 to the client
    c. Increase the counter for the number of queries
    d. Keep going until server sends "done" → while loop
        i. For each buffer the server sends, save it to the pokemon linked list
6. If user input is 2
    a. Ask user for file name
        i. If it doesn't work, show error and ask again
    b. Create file using file name
    c. Go though Pokémon linked list and for each node, write it out in the file
    d. Save file name in filename linked list
7. If user input is 3
    a. Go though the file linked list and print out all of the file names
    b. Print number of queries
    c. Free up the memory from the linked lists (pokemon and file names)

d. Send stop to the server
e. Shut down the client

Header Files

- <stdio.h> → Standard C library for C code
- <stdlib.h> → C keywords → sizeof() for the size of the char array/string
- <string.h> → String modification
- <unistd.h> → multiprocessing
- <sys/socket.h> → Creating server sockets and sockets
- <netinet/in.h> → Creating server ports and sockets
- <arpa/inet.h> → Creating server ports and sockets

NFR1 – Performance

- When searching for the type 1 from the file, it should run as O(n) time because its just looking for the 3rd term in the csv file. So we basically wait until the server finished looking up the entire file then show back the menu up again.

NFR2 – Maintainability

- Included a makefile that compiles and creates the exicutables from the code from the client and server side so we just need to type in make in the terminal so we can run the code‼