Name: Asad Ansari

Carleton ID: 101216870

Pseudocode:

File assignment4PPC.c (client side code):

```
Including files: stdio.h, stdlib.h, unistd.h, string.h, sys/socket.h, netinet/in.h, arpa/inet.h, pokemon.h
Char buffer[20];
Char a[20];
//allocating memory for variable c1
//allocating memory for variable c1->pokemon_array
c1->num_pokemon = 0
pthread_t t2
int querie = 0
int numOutput = 0
char OutputLine[100] = “”
//creating socket
//setting up address for client
While(1):
    //print options and collect input into variable a
    If(input == “a”):
        buffer = Input(“enter type pokemon”)
        //sends input to server
        //client will collect number of pokemons from type search and receive each line from server
        //client will convert the line of pokemon collected and put it into own pokemon_array
    If(input == “b”):
        Output = Input(“Enter name of output file”)
        //using while loop, checks to see if name is valid
        //if name valid then breaks loop and uses thread to run write pokemon function
        If(input == “c” || input == “d”):
                //frees everything allocated in memory and closes client program
        Else:
                Input(“Enter valid option”)
End of assignment4PPC.c file
File assignment4PPS.c (server side code):
Including files: stdio.h, stdlib.h, unistd.h, string.h, sys/socket.h, netinet/in.h, ctype.h, pokemon.h
Char buffer[20] = “”
Char fileName
Pthread_t t1
//allocating memory for count_t struct
C1->num_pokemon = 0
fileName = Input(“Enter name of pokemon file”)
//while loop to check is file name is valid, other wise user is to enter name of file again
//creating server socket
//setup server address
//bind the server socket
//waiting for clients
Buffer = //whatever received from client
If(buffer != “c” && buffer != “d”):
        //creating semaphore
        //using thread to run read_pokemon
        //read_pokemon will read the input file and create structures of Pokemon into pokemon_array
        //destroying semaphore
        //sending num_pokemon to client then sending each individual line of pokemon found from
        input file to client
If(buffer == “c”):
        //close input file and free everything
        Num_pokemon = 0
```

```
If(buffer == “d”):
        //close input file and free everything and break out of main while loop
//the program exits and server stops
```

Ive used the libraries stdio.h, stdlib.h, unistd.h, string.h, sys/socket.h, netinet/in.h, arpa/inet.h, pokemon.h for the server code as stdio and stdlib is to collect input and output of program. string.h is used for string functions. Sys/socket is used for server to access socket and netinet and arpa/inet also helps with create server capabilities. Client has all the same h files including ctype.h which is for basic functions such as isDigit(). For functional requirements, the client first gets input of what option to select. If user selects option type search, it will send search to server and server will collect all the pokemon of that type and store it in an array. The server will then break each structure in that array back into a line and send that to the client to store in its own structure. The structure array is then accessed if user selects option 2 save file. When save file is selected, each structure will be written into its own line and printed in desired output file. If user is to pick option 3, quit the client, then everything which was allocated will be free’d and the client program will quit while the server is still running. Option 4 is where the user wants to quit the client program and the server program all together. Everything will be free’d and server and client program will stop. Regarding the performance part of the non-functional requirements, the query program remains responsive to the user input regardless of number of queries that have been submitted by the user, while queries are executing, and while writing the results to disk. From maintainability, the server and the client programs are both on separate c files and there is a header file created called pokemon where all my functions are. A Makefile is also created for easy compilation of code.