Assignment 2: IPEssentials

Goals of this Assignment

- Learn to create simple and nested loops
- Learn to create and use user-defined functions
- Learn to define and use one-dimensional arrays and strings
- Learn to read given requirements and convert them to C code
- Use problem solving skills to solve a given problem

1.0 Description

An IPv4 address is a unique identifier assigned to each device connected to a network that uses the Internet Protocol for communication. It is a 32-bit number typically represented in a human-readable format as four decimal numbers separated by dots (e.g., 192.168.0.1). These 4 decimal numbers are also known as octets. For example, in the IPv4 address 192.168.0.1, the numbers 192, 168, 0, and 1 are the four octets. Each octet can represent a value from 0 to 255, allowing for a wide range of unique addresses within the IPv4 system.

2.0 The task

Your task for this assignment is divided into 3 parts, each part is explained below.

First things first

You are given a header file called givenA1.h – this file contains the proper function prototypes, constants, and structures required for this assignment, also detailed below. Note that this file has an extension of h.

There are two source (2) files that you must submit:

(1) You will write a .c file, lastnameFirstnameA2.c containing the function implementations detailed below
(2) You will write a .c file, lastnameFirstnameA2Main.c with the main function

Include the given file givenA2.h in your C files (i.e., add the following as the first line in both the c files)

```
#include “givenA2.h”
```

To compile them together, use the following gcc command:

```
gcc -std=c99 -Wall lastnameFirstnameA2.c lastnameFirstnameA2Main.c
```

More on this can be found in the section on "Submission Instructions".

You are also free to create additional helper functions, though if they are used by the assignment functions below, include them in the appropriate files so they are accessible to us during grading.

Part I: You must write function definitions for 3 main tasks given below in lastnameFirstnameA2.c file.

Task 1:
Create a function called readIPAddress() that prompts the user to enter a single IPv4 address. Note that IPv4 and IP address mean the same for this assignment. The user must enter 4 integer numbers (i.e., 4 octets), each number must be in the range of 0 to 255. Users are repeatedly prompted to enter a valid octet if they don't. This must be validated for all 4 octets. The function then collects the 4 octets separating them by dots into a string that resembles an IP address. For example, if the user enters the 4 octets as 75, 71, 1, 0, then the string to represent the IP address stores "75.71.1.0" without quotes.

Correction:
Note that the 1st integer, i.e., the first octet MUST be in the range of 0 - 127 (and not 0 to 255. The rest of the octets are in the range of 0 to 255.

Prototype:

void readIPAddress (char ipAddress []);

Example scenario:

```
Octet#1 - Enter a number - must be between 0 and 127: -2

Octet#1 - Enter a number - must be between 0 and 127: 129

Octet#1 - Enter a number - must be between 0 and 127: 125

Octet#2 - Enter a number - must be between 0 and 255: 255

Octet#3 - Enter a number - must be between 0 and 255: 1

Octet#4 - Enter a number - must be between 0 and 255: 0


IP Addr: 125.255.1.0
```

Note that the function does not print the IP address – assume that it is displayed after the function is called in main for testing purposes.

Task 2:

In this task, you have to write a function that take2 inputs, a string that represents a valid IP address and an integer indicating the length of the IP address, and returns the decimal equivalent of the IP address as a long int. It also outputs the total number of digits in the resulting long int via call-by-reference.

To convert the IP address to a long int, the function takes each octet, converts it to an integer, finds the binary equivalent of the octet and stores it using 8 bits (note that a bit is either 1 or 0). It then combines the 8-bits of each octet into 32-bits, and then converts these 32-bits to its decimal equivalent as a long int. An example is given below.

Prototype:

```
long int convertIPToLongNumber (  char ipAddress [],
                                  int lengthIPAddr,
                                  int * numDigits);
```

For this assignment, we will assume that an integer is stored in the memory as 8 bits and a long int is stored as 32 bits.

Example: if the given IP address given to the function is

198.78.1.0

It is stored using 32 bits as
11000110010011100000000100000000

(How do we get the above 32 bits: Note that integer 198 in binary form is 11000110; integer 78 is 01001110, integer 1 is 00000001 and integer 0 is 00000000. Combining the above, we get the 32 bits shown above).

The above 32-bit binary representation is then converted to its decimal equivalent and stored as a long int is

3327000832

The number of digits in this long int is 10.

Task 3:

In this task, you have to classify a given IP address and return its class as a char (A, B, C, D or E) as described below.

IP addresses are categorized into classes (A, B, C, D, E), where classes A, B, and C are commonly used for general networking, while classes D and E are reserved for multicast and experimental purposes, respectively. A brief explanation of these classes is given below:

1. **Class A:**
   - The second octet ranges from 0 to 126.
   - Class A addresses are used for large networks.
2. **Class B:**
   - The second octet ranges from 128 to 191.
   - Class B addresses are used for medium-sized networks.
3. **Class C:**
   - The second octet ranges from 192 to 223.
   - Class C addresses are used for small networks.
4. **Class D:**
   - The second octet ranges from 224 to 239.
   - Class D addresses are used for multicast groups.
5. **Class E:**
   - The second octet ranges from 240 to 255.
   - Class E addresses are reserved for experimental purposes.

Note that in reality, such a classification is based off the first octet of an IP address, but for this assignment, you must base it off the second octet.

Prototype:

```
char classifyIPAddress (char ipAddress []);
```

Example:
The second octet of 198.23.254.1 is 23, and therefore the class of this IP address is A.

Part II: You must write and use the following helper functions for the required tasks given in part I. All helper functions must also be written in lastnameFirstnameA2.c file

```
int  countDig (int);

void convertToBinary (int octet, int octetBinary [8]);

void combineAllOctets (int octetBinary[8], int pos,
                                    int binaryAllOctets [32]);

int  convertBinaryToDecimal (int binaryAllOctets [32]);
```

Part III: Write lastnameFirstnameA2Main.c to test each function implemented for parts I and II.

3.0 Submission Instructions:

- Submit both C files containing your function definitions and main. To submit, upload your C files to the submission box for A2 on Courselink. Remember to name your files as lastnameFirstnameA2.c (For example, if Ritu is the first name and Chaturvedi is the last name, the file would be called chaturvediRituA2.c) and lastnameFirstnameA2Main.c. Incorrect file name will result in penalty.
- Incorrect format of submitted files will result in automatic zero. (Must be a valid .c file)
- The program you submit must compile with no warnings and run successfully for full marks. You get a zero if your program doesn't compile. There is also a penalty for warnings (5% for each unique warning).
- Penalties will occur for missing style, comments, header comments etc.
- DO NOT use global variables. Use of any global variables will result in automatic zero.
- DO NOT use goto statements. Use of any goto statements will result in automatic zero.
- DO NOT use concepts such as two-dimensional arrays and strings, structs and files – use of any of these will result in penalty.
- Use the template given below for header comment in each c file you submit.
  /!\ Note: The file name, student name and email ID must be changed per student.

```
/***********************chaturvediRituA2.c*************

     Student Name: Ritu Chaturvedi Email Id: ritu

    Due Date: October …                    Course Name: CIS 1300

    I have exclusive control over this submission via my password.
    By including this statement in this header comment, I certify that:

    1) I have read and understood the University policy on academic integrity. 2) I
    have completed the Computing with Integrity Tutorial on Moodle; and 3) I have
    achieved at least 80% in the Computing with Integrity Self Test.

    I assert that this work is my own. I have appropriately acknowledged any and
    all material that I have used, whether directly quoted or paraphrased.
    Furthermore, I certify that this assignment was prepared by me specifically for
    this course.

*********************************************************/
```

The program file must contain instructions for the TA on how to compile and run your program in a header comment.

/!\ Note: The file name must be changed per student.

```
/*********************************************************
Compiling the program
The program should be compiled using the following flags: -std=c99 -Wall

compiling:
gcc -std=c99 -Wall chaturvediRituA2.c chaturvediRituA2Main.c

Running: ./a.out
OR

gcc -std=c99 -Wall chaturvediRituA2.c chaturvediRituA2Main.c -o assn2

Running the Program: ./assn2
*********************************************************/
```