**Chapter 4 Programming Assignment**

2 Points

Problem: You are new to the Linux world and want to make sure that the input received is in correct octal format for the chmod permissions command. The octal number system using values 0 to 7. The chmod command accepts three octal numbers to set the permission values.

Procedure: read in the 3 digits from the user for the octal permission command, check if each digit is within the range and then state if those three numbers make a valid octal number for the chmod command.

Example: 0 6 3 yes, 063 is a valid octal number for the chmod command

9 0 2 no, 902 is not a valid octal number

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3 Points

Problem: You are working and internship for the DNS, your boss wants you to figure out if the numbers requested make a valid IP address. A valid IP address is made up of four groupings (called octets) of three digits, each grouping must be between 0 and 255. You need to state if the given four octets make a valid IP address; if it does not, then you need to state which octet(s) is invalid.

Procedure: read in four octets, check if the octets will make a valid IP address, if it does not, you will also have to check each octet and state which octet(s) is not valid.

Example: Please enter the first octet: 112

Please enter the second octet: 200

Please enter the third octet: 0

Please enter the fourth octet: 254

Valid IP Address: 112.200.0.254

Please enter the first octet: 312

Please enter the second octet: 120

Please enter the third octet: 50

Please enter the fourth octet: 357

Invalid IP Address: 312.120.50.357; 1st octet, 312 is incorrect 4th octet is incorrect

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4 points

Problem: You have the task to parse data that has been entered by users and are tired of dealing with runtime errors of bad typing by the users. So, you want to add a bit of code to help with the following:

1. the error of excess spaces that are entered around a word when a word is requested as input
2. the error a single digit not being a number when a number is requested as input

Procedure: read in a word, eliminate the spaces before and after the word

Read in a single “number,” check if it is a digit, then parse the digit to an int value

Note, there is not output associated with this, just internal memory storage correctly, unless the input cannot be accepted.

Example: Please enter a word: realization

\_\_\_realization\_\_\_\_ is stored as: realization

Please enter a whole number between 0 and 9: 8

8 is read in and then stored in memory as an int value

Please enter a word: comprehension

\_comprehension is stored as: comprehension

Please enter a whole number between 0 and 9: a

a is not a digit

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6 points

Problem: You want to write a program that mixes words up – where the back half of the word is at the beginning of a word, thus the front half of the word is at the end of the word.

Procedure: read in a word, get the length of the word, create 2 substrings – a front half and a back half, and make a new word with the back half and concatenating the front half on.

Example: hello becomes llohe

encryption becomes ptionencry