William George
11/5/2014
CS 161 Assignment Week 6
Arrays and Pointers

Understanding:

The purpose of this week's projects and exercises is to get a better understanding of pointers and arrays. Both were introduced this week in the lectures and in the reading. For the project the goal is to make a program that will accept a word input from the user then give the user a few choices for of commands to manipulate the word. These commands include reversing the entire word input, shifting the word left then adding any extra characters to the end, shifting the word right then adding any extra characters to the front and quitting the program. With the commands (besides quit of course) the user should be able to manipulate the string as many times as he/she wishes until they decide to quit.

From a developer standpoint the commands will call three different functions each performing one of the manipulation types on a c-string variable which acts like an array. Because this week focuses on both arrays and pointers, pointer arithmetic will be used to manipulate the word. Also a couple of the commands (namely the left and right shift) will be saying two different things, to shift in a direction and the number of spaces to shift, as such we need to change the input so that it can be separated into parts. My thoughts are to use the first character to determine what command to give, then turn the string into c-string so you can easily extract the second character. This will also help with validating the command input.

From the looks of it this project will be very challenging especially with the left and right shift functions as for each you will have to take a letter off per each iteration of a loop then move the c-string over a space, then add the letter taken off to the other side.

Design:

Directives: *iostream*
string
cstdlib

Functions: *getMenu – no parameters*
isNum- string number parameter
reverse – c-string, c-string length parameters
leftShift – c-string, c-string length, degree of shift parameters
rightShift– c-string, c-string length, degree of shift parameters

Main Function
Variables: *string user word input*
String user command input
c-string SIZE = 30

```
                char c-string word input
                char command input c-string
                char word shift parameter
                int word shift parameter
                int word length parameter

        output: user input word
        input: string user word
        convert string to c-string
        find c-string length
        clear screen
        Call getMenu function

        Do- while command is not quit,, Quit or QUIT
            {
            Clear buffer
            Output: enter command
            If command = rev
                Call reverse function
                Output new c-string
            Else if command starts with L
                Change command to c-string
                Extract second character and validate as number input
                While not number ask user for number
                Change character to integer → set to shift
                Call left shift function with user string, word length and shift parameters
                Output new c-style string
            Else if command starts with R
                Change command to c-string
                Extract second character and validate as number input
                While not number ask user for number
                Change character to integer → set to shift
                Call right shift function with user string, word length and shift parameters
                Output new c-style string
            Else if command is quit, QUIT or Quit
                Say thank you and fall through
            Else command is invalid, reenter valid command
            }
        End program

    getMenu function
        output → to reverse word enter command rev
        output → to shift the word left enter Lx with x being the number of spaces you'd like to shift it
        output → to shift the word right enter Rx with x being the number of spaces you'd like to shift it
```

```
          output→ to quit enter "quit"

isNum function
          if character is not greater than and less than or equal to nine then return false, else return true

reverse function
          declare variables for first and last characters as well as one for temporary
          set variable to last character
          for ½ length of the word
                    swap beginning and end characters
                    increment beginning up
                    decrement last character down
end function

right shift function
          set variable for temporary values
          loop until the end of user specified shift
                    take the character from the end of the word
                    shift all characters one space right
                    add the character from end to the beginning of the word
end function

left shift function
          set variable for temporary values
          loop until end of user specified shift
                    take the character from the front of the word
                    shift all characters left one space
                    add character from beginning to end of word
end function
```

Testing:

User Word	**Command**	**Expected Output**	**Actual Output**
Wordshift	**rev**	**tfihsdroW**	**Expected**
Wordshift	**rev**	**Wordshift**	**Expected**
WordShift	**R2**	**ftWordshi**	**Expected**
Wordshift	**L2**	**rdshiftWo**	**Expected**
Wordshift	**quit**	**Thank you!**	**expected**
Wordshift	**r2**	**Incorrect input try again**	**Expected**
Wordshift	**Rd**	**Incorrect input please enter how far you'd like to shift the word**	**Expected**
Word shift	**Rev**	**drow**	**Expected**
Word shift	**REV**	**drow**	**Expected**

Wordshift	**3**	**Incorrect input try again**	**Expected**
Wordshift	**red**	**Incorrect input try again**	**Expected**
Wordshift	**Red**	**Incorrect input please enter how far you'd like to shift the word**	**Expected**

Reflection:

This week was challenging just from a logic standpoint. While the arrays made things easier the mix of the arrays and pointers made it all confusing again. I understand the two are related, but the pointers are a more complicated subject than I thought. I had no problem with building the main function in the project beyond the decision to use else if statements rather than a switch case with each condition saved to some integer, once I figured out the left and right shift words need to be c-strings so that I can separate the numeric character from the alphabetic character it all fell into place. The functions however were a different story.

I struggled mightily with the right shift function and left shift functions. The reverse function was relatively easy, but with the other two I what I needed to do per each iteration of the loop but was unsure how to do it. I also did not know how I wanted to set up the nested loop. What helped me most was to change my thought process and start thinking about how to manipulate the arrays rather than the pointers. Once I got an understanding of how to manipulate the arrays I then translated it in terms of the pointers…so to move left instead of the language being array [i] = array[i + 1], I changed it to *(array +i) =*(arrayPtr +i+1) and the problem got simpler. After figuring out the left shift I spent way too much time trying to figure out how to move right. Apparently just reversing the function was too easy for me to think it was the correct means. For me the challenging problems are the most satisfying when completed, but sometimes my level of persistence gets in the way and when I should just walk away from a problem I tend to just look at it and dwell on it until I can complete it. I saw that this week with these problems and something that should have taken a couple hours took an entire day.