Amy Zhang

CSCI-313

HW2- Question 6: Chuck Norris Problem

To begin, in the main function, I created an ifstream object called inputFile that will be used to read in text. In addition, I created another string variable called line that will store each line of text read from the “Chuck Norris Jokes.txt” file. Next, inputFile.open(“Chuck Norris Jokes.txt”) simply just opens the file so information can be read. Moving on with the code, there is an if statement that checks whether the file is open or not. If the file doesn’t open, it’s likely that the file was not found. Now let's assume that the file opens, in that case we will reach the outer while loop, which will read each line in the file and store it into the variable line by using the getline function. Next, I created a stringstream object called wordStream that’s used to tokenize each word. The term tokenize means to split a string into multiple substrings by using delimiters or characters that separate text strings such as commas, semicolons, etc. In my case, I didn’t specify any delimiters. Therefore, it uses the default delimiter, which is the whitespace. I created another object word of type string that will store each word. In addition to the outer while loop, I included an inner while loop that will read each word in the line and store it in variable word. Within the inner while loop, I declared an int variable size and initialized it to the length of the word that’s being read in, which will be used later in the reverse function. This following line of code: cout << reverse(word, size) << " "; will call the reverse function and it will reverse each word in the line and print the result.

Moving forward, I created a reverse function that takes in two parameters and returns a string, which is as follows: string reverse(string str, int size). Within the reverse function, I created an empty stack called s used to store characters. Then, I used a for loop to push elements

from input str into stack s. By doing so, stack s will contain all the words stored in str, and input str will be empty. Next, I used an if statement to check if the last character is a punctuation. The ispunct function is used specifically to check if the last character is a punctuation or not. Assuming that the last character is not a punctuation, then it will get pushed into the stack, otherwise it will be skipped and not be added to the stack. For instance, let's pretend that the input str is "word.", then after reversing, it should be "drow." instead of ".drow". The point is that the punctuation/period should be at the end of the reversed word and not the beginning. Continuing onwards, I declared an int variable k and initialized it to -1. This variable k will serve as the index for input str within the upcoming while loop. Since I coded as follows: str[++k], the k needs to be initialized to -1 because the index must start at 0. Next, I used a while loop to go through the stack and pop each element until it's empty and it will assign the values from top to bottom of stack back to input str. The updated input str will return back to the main function at the location of where it was called and output all the reversed words. Now back at our main function, we encounter the code inputFile.close(); which just simply closes the file we were working with. The last block of code is the else statement which is continuing the if statement before it. If the if statement block of code doesn't execute, then the else statement will execute and let the user know that the file did not open and no reversing was done.