Question 6

This comedic problem was fun to solve but it also makes it very easy to understand how simple a stack is. It really is just a pop and push data structure. This problems core is in the function “reverseWordsInString”. The function passes in a string by reference, and the size of the string. A temporary stack is constructed and I push on a empty character because it was the only hard coded solution I could think of. I had an issue where the first two words that were outputted were always concatenated. I declare and initialize two variables to 0. The variable i will traverse the string while k will hold the current position in the stack. The while loop begins and will run until it reaches the end of the string that is passed through. Using the built in C++ function “isspace”, I check if the current character is a space and then start placing the characters from the stack onto the string and then pop them after because we want to reverse each word and not the entire string. Once we pop at the end of each loop, we are able to start with a fresh empty stack at the beginning of each word which are signaled by the spaces. If the current character is not a space, then I push the character onto the stack. Lines 29 through 3 2 are there for the last word. There isn’t a space after the last word to trigger the inner while loop, lines 29-32 take care of this issue buy popping the stack one last time and placing the words onto the end of the reversed string. In my honest opinion, I did not have much difficulty solving this problem. The biggest issue was the concatenation of the first two words, and it does seem silly. I kept trying to fix the algorithm while there was a simple solution the entire time. Sometimes while coding, more often than not, we tend to overthink and get lost in the problem which lead to unclear ideas. This was a fun problem and Chuck Norris would appreciate it.