Dean Pektas CS313 11/9/20

Question 2: The program works to evaluate prefix expressions and calculate them. It starts by creating an empty stack that will be used later in the program to push each character in the given expression into itself. The “evaluateExpression” function takes in an expression of type string. The for loop contained in the function will go through each index of the expression, starting from the end of it and will make its way to the beginning. A decision is made for each index to see if that index is a digit, and if so, will be pushed into the stack as a digit. This is handled through the “isOperand” function which returns any number that's not zero if it is a digit. Zero is considered not a digit. As the for loop continues, if it is determined that the specified index is not a digit, it is assumed that it is an operator. Therefore, the two available digits that came before the operator will be stored in variables “op1” and “op2”, and they both will be popped from the stack. These values along with the operator that was just found will be passed into another function called “evaluateOpr”. This function uses a switch statement that, based on what the operator is, will perform the appropriate operation on both op1 and op2, and push the resulting value into the stack. Once the entire expression has been evaluated, the top of the stack will be returned. The main function allows for the user to type whatever expression they would like, and the rest of the functions will evaluate it.

Question 6: The program works to take a string of words and place the words in reverse order. It starts by creating an empty stack as well as an empty string variable called “plate” that will be used to temporarily store portions of the expression into it. The “evaluateExpression” function takes in an expression of type string. The for loop contained in the function will go through each index of the expression, starting from the beginning, unlike the previous question. For each index, a decision is made as to whether or not the index is empty. If so, it will call a function called “emptyTemp” where it will push the temporary plate variable onto the stack and will erase whatever was already in the variable. If it is the case that the index is not empty, it will call a function called “writeToPlate”, passing in the contents of that index. This function adds what’s currently in the plate variable plus the content from the index that was passed in. It concatenates the next piece of the expression. After completely evaluating the expression, the plate variable gets pushed into the stack with the new, reversed expression. A while loop is then performed if the stack was actually filled with information, where the plate variable will be set to the top of the stack (which is the end of the expression, and print out the contents. Each time this is done, that part of the stack is popped and will continue until the stack is empty. The main function carries a sample string that, when passed through to the “evaluateExpression” function, will output the reverse in the terminal.