Daniel Yoh

CSCI 313-22

#4: Create your own stack functions to do the following:

a. List all the elements in the stack

b. Iterate through the stack and change one of the values based on its position

I added two functions, stackPrint and stackChange, to the original myStack.h header file. They both take in a reference to an array stack, but the second one that changes a value also takes in a position, a new value, and the size of the stack.

Both functions operate under similar principles. They both use a temporary stack and temporary integer, along with a pair of loops to go through the stack and copy elements into the temporary stack, pop elements from the original, then eventually transfer the elements back into the original stack from the temporary stack. For stackPrint, a counter is kept so that the list number of each element can be displayed when the element is printed.

In stackPrint, two while loops are used. The first loop is active for as long as the original stack isn't empty. It pushes its elements into the temporary stack, then pops them from the original stack. Then the second while loop goes on until temp is empty. It pops elements from temp, and begins printing them while pushing the top back into the original stack. To print the elements of a stack in order, we first need to delete the elements so we can go to the beginning, then start printing from there while refilling its elements back in order.

For stackChange, two for loops are used. The first keeps iterating until it reaches the position where the element is to be changed. The top of the original stack is pushed onto the temporary stack, then the original is popped. After the loop ends, the new value is pushed into the original stack, which will be at the new top. Then, like in changePrint, the original elements are refilled into the stack through the temporary stack. but with a for loop. The same principle as in the while loops is necessary here; to replace an element, we need to keep deleting elements until we reach the correct position, then we can place it in and refill the stack.

I just now realized that I might have been able to use a while loop here like stackChange instead of a for loop, which probably would've saved me a good amount of time trying to figure out the right conditions for the for loop.

There is a main program to test these two functions. The program begins with including <iostream> and "myStack.h". An integer is given a size, then a stack is declared and initialized with that size. A for loop fills it with elements matching i, and the print function is tested. Then the stack is changed; the element at position 4 is replaced by a 20, then the new stack is printed to show its effect.