DSAP-HW7

b06303077 Yu-Jo Chiang

April 2020

1. Problem 1

Listing 1: Print out right direction

```
1     // Reads an input line,
2     //recognizing the character '<-' as a backspace
3     //that erases the previously typed character.
4     //Returns a stack of the corrected characters read.
5
6     readAndCorrect(): Stack
7     Initialize aStack as an empty stack
8     Initialize bStack as an empty stack
9     Read newChar
10    while newChar is not the end-of-line symbol
11        if newChar is not a '<-'
12            aStack.push(newChar)
13        else if aStack.isEmpty() is false
14            aStack.pop()
15        Read newChar
16    return aStack
17
18
19    //move to bStack
20    displayBackward(aStack: Stack)
21    while aStack.isEmpty() is false
22        newChar = aStack.peek()
23        bStack.push(newChar)
24        aStack.pop()
25
26    //print in  right direection
27    displayForward(bStack: Stack)
28    while bStack.isEmpty() is false
29        newChar = bStack.peek()
30        bStack.pop()
31        Write newChar
```

2. Problem 2

Listing 2: Check correct brackets

```
1     //Reads an input line recognize all the brackets
2     //write the brackets into a stack to check the orders and numbers
3
4     readStack():stack
5         Initialize aStack as an empty stack
6         Initialize bStack as an empty stack
7         Initialize cStack as an empty stack
8         read token
9         while token is not EOF
10            if token == "{},[].()"
11                aStack.push(token)
12                bStack.push(token)
13            read token
14        return aStack
15        return bStack
16
17    //count number
18        while aStack is.empty() is false
19            aStack.pop()
20            count ++
21
22    //check order
23        if(count/2 != 0)
24            return false
25        else
26            while cnt < count/2
27                aStack.pop();
28                cnt++
29            while aStack is.empty() is false
30                newToken = aStack.peek()
31                aStack.pop()
32                cStack.push(newToken)
33            while cStack is.empty() is false
34                bToken = bStack.peek()
35                bStack.pop()
36                cToken = cStack.peek()
37                cStack.pop()
38                if(cToken!= bToken)
39                    return false
40                    break;
41        return true
```

3. Problem 3

- stage 1 : for the first while loop it takes (n) to work, and for the second loop it takes at most 2 times to compile, so the big O $\rightarrow$ 2n
- stage 2: left with 1 $\rightarrow$ 1
- bigO : n