DSA
PS2, due Wed Aug 25 8:00 PM

See notes for review of stacks.

Please solve these problems using a stack thought process

1. Read chapters 6.1 of the textbook

2. Remove all adjacent duplicates. Given a string *S* of lowercase letters, a duplicated removal consists of choosing two adjacent and equal letters and removing them.

   ```
   input:  s = “abbaca”
   output: “ca”
   We could remove “bb” because the letters are adjacent and equal. After
   we do that, we get “aaca”, where we can again remove “aa”, until we are
   left with “ca”.

   input:  s = “azxxzy”
   output: “ay”
   We remove “xx” to get to “azzy”, then remove “zz” to get to “ay”.
   ```

3. Given a string *S* of lower *and* upper case letters, return a good string, defined as <u>not having</u>:

- `0 <= i <= s.length - 2`
- `s[i] is a lower-case letter and s[i+1] is the same letter as upper-case or vice versa`

```
input:  s = “giIiraffe”
output: “giraffe”
If you choose i=1 or i=2, you still remove “iI”, s[1:2] or “Ii” s[2:3],
either way you end up with giraffe.

input:  s = “abBAcC”
output: “”
You can get this a number of ways. You can start by removing “cC” or
“bB”, then you will remove “aA” or “cC” or “bB”, so on and so forth,
always ending up with “”.

input:  s = “s”
output: “s”
There’s no sS or Ss situation here we just have one letter so we return
it.
```

4. Given an array `target` and an integer `n`, build the `target` array using the following operations that will pick out the elements of target from the range 1 up to n+1. Say if I wanted a target of [1, 3] with n=3, I am asking to iterate over range(1, 3+1), or 1 2 3 4, and output the stack operations that will give me the array [1, 3].

- **Push**: read a new element from the beginning list and push it into an array.
- **Pop**: delete the last element of the array.
- If the target array is already built, stop reading more elements.

```
input:  target = [1,3], n=3
output: [“Push”, “Push”, “Pop”, “Push”]

Read number 1 and automatically push in the array -> [1]
Read number 2 and automatically push in the array then Pop it -> [1]
Read number 3 and automatically push it in the array -> [1, 3]

input:  target = [1,2], n=4
output: [“Push”, “Push”]

The range we iterate over is range(1,n+1), or range(1,5), encompassing
the numbers 1 2 3 4 5. Of this number list, we want as our target the
array containing [1,2]. This is just the first two numbers of 1 2 3 4
5, which just means we will want to push the first two elements.
```

5. The next greater element of some element `x` in an array is the first greater element that is to the right of `x` in some array. You are given two distinct 0-indexed arrays `nums1` and `nums2`, where `nums1` is a subset of `nums2`. Everything in `nums1` will be in `nums2`, but `nums2` will have some more stuff in it. For each `0 <= i < nums1.length` , find the index `j` such `that nums1[i] == nums2[j]` and determine the next greater element of `nums2[j]` in `nums2`. If there is no greater element, then give -1. Return an array *ans* of length `nums1.length` such that `ans[i]` is the next greater element as described above.

```
input:  nums1 = [4,1,2] nums2 = [1,3,4,2]
output: [-1,3,-1]

The next greater element of 4 is -1 since nums2 has no larger numbers
than 4 after 4 (there’s just 2). The next greater element of 1 is 3,
since 3 is the next largest (the first greater element) to the right of
1 in the array nums2. The next greater element of 2 is -1, since
nothing lies to the right of 2 in nums2.


input:  nums1 = [2,4] nums2 = [1,2,3,4]
output: [3,-1]

The next greater element of 2 is 3, since 3 is the first greater
element that is to the right of 2 in nums2. The next greater element of
4 is -1, since nothing lies to the right of 4 in nums2.
```