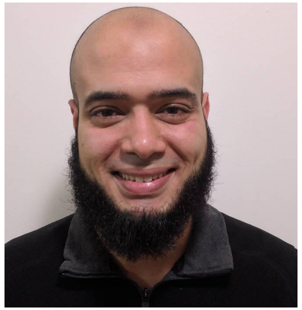
Data Structures
Stack Homework #1

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Problem #1: Another Stack array design

- One of the students implement his stack with the following design choices
  - Instead of top, used int added_elements to represent number of added elements so far
  - Instead of assert, push, pop, peek ⇒ returns bool to indicate success / failure
  - If we push 10, 20, 30 in stack, the array content is: {30, 20, 10}
    - That is, array[0] represents the top
- Change the lecture stack code to reflect these changes
- Discuss the time complexity of the methods
  - Discuss how even minor design choices affect our data structure!
- Change the main() function to reuse the new interface of the code

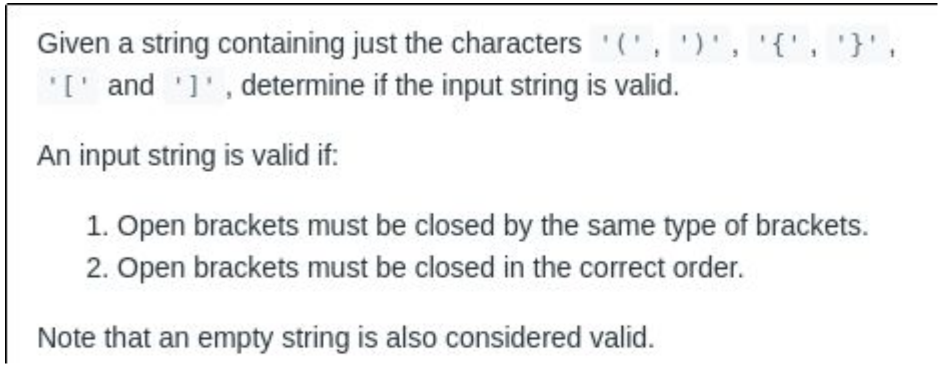
Problem #2: Reverse subwords

- Develop: string **reverse_subwords**(string line)
- It takes a string of spaces, e.g. "abc d efg xy"
- Then reverse each subword ⇒ “cba d gfe yx“
- Find a stack-based idea

Problem #3: Reverse a number using stack

- Implement a method that takes number >= 0 and reverse its digits using a stack
- int reverse_num(int num)
- E.g. Input: 1234, Output: 4321
- Find a stack-based idea

Problem #4: Valid Parentheses


Given a string containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

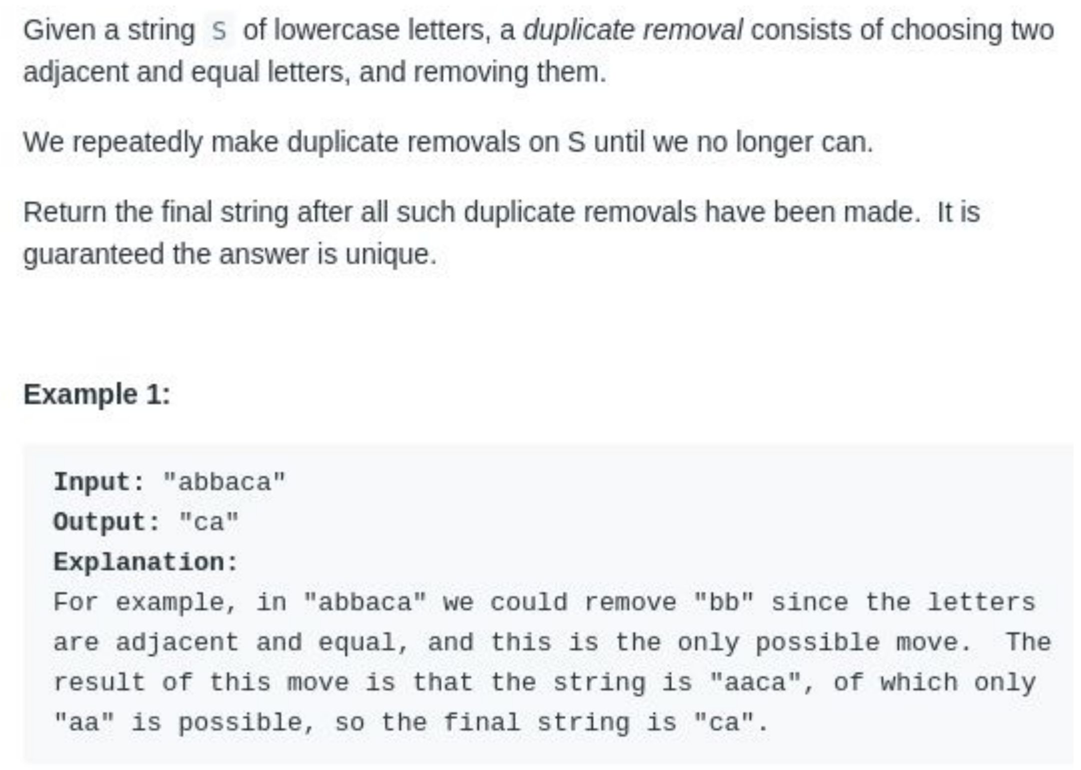
An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

Note that an empty string is also considered valid.

- Develop: bool isValid(string str)
- Valid: (), ()(), (()()), {}{}, (((()))), ([]), ()[]{}, {[]}
- Invalid: (], ()), (][), (], ([)]

Problem #5: Remove All Adjacent Duplicates In String



Given a string `S` of lowercase letters, a *duplicate removal* consists of choosing two adjacent and equal letters, and removing them.

We repeatedly make duplicate removals on S until we no longer can.

Return the final string after all such duplicate removals have been made. It is guaranteed the answer is unique.

Example 1:

```
Input: "abbaca"
Output: "ca"
Explanation:
For example, in "abbaca" we could remove "bb" since the letters
are adjacent and equal, and this is the only possible move.  The
result of this move is that the string is "aaca", of which only
"aa" is possible, so the final string is "ca".
```

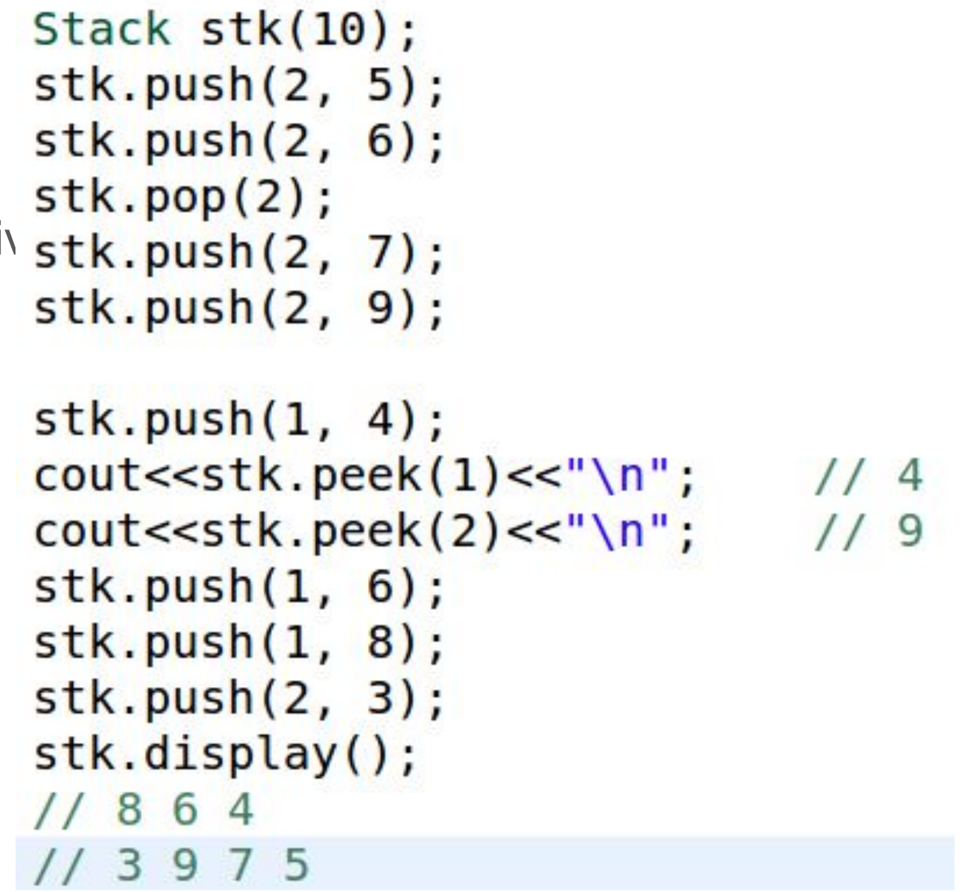
- Use a stack
- string removeDuplicates(string S)

Problem #6: Two stacks

- Change our class to be able to perform the stack operations but simulating 2 stacks in same time. Consider the following constraints
  - Use only the same single array
  - All operations should have the same old time/memory complexity
  - Each stack can use as much as possible of available array
    - For example, assume we created stack of 100 elements
      - The first stack used 20 elements so far
      - Then the second stack can use up to 80 elements and so on
- Change functions similar to following: void push(int id, int x)
  - Id either 1 or 2 to refer to the stack itself

Problem #6: Two stacks

- We create a stack of 10 elements
- Then we can perform operations, gi\
  the id



```
Stack stk(10);
stk.push(2, 5);
stk.push(2, 6);
stk.pop(2);
stk.push(2, 7);
stk.push(2, 9);

stk.push(1, 4);
cout<<stk.peek(1)<<"\n";     // 4
cout<<stk.peek(2)<<"\n";     // 9
stk.push(1, 6);
stk.push(1, 8);
stk.push(2, 3);
stk.display();
// 8 6 4
// 3 9 7 5
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."