Interview questions for BuildHer Campaign.

Easy 1

Given a string `s` consisting of words and spaces, return *the length of the **last** word in the string.*
A **word** is a maximal
substring consisting of non-space characters only.
Constraints:

- `1 <= s.length <= 104`
- `s` consists of only English letters and spaces `' '`.
- There will be at least one word in `s`.

CODE:

```
def length_of_last_word(s):
    words = s.split()
    if not words:
        return 0
    return len(words[-1])

# Example usage:
s = "Hello World"
result = length_of_last_word(s)
print(result)
```

OUTPUT: 5

Medium 1

Given a binary search tree (BST), find the lowest common ancestor (LCA) node of two given nodes in the BST.

According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes `p` and `q` as the lowest node in `T` that has both `p` and `q` as descendants (where we allow **a node to be a descendant of itself**).”

Constraints:

- The number of nodes in the tree is in the range `[2, 105]`.
- `-109 <= Node.val <= 109`
- All `Node.val` are **unique**.
- `p != q`
- `p` and `q` will exist in the BST.

CODE:

```
class TreeNode:

    def __init__(self, val=0, left=None, right=None):

        self.val = val

        self.left = left

        self.right = right

def lowest_common_ancestor(root, p, q):

      if p.val > q.val:

        p, q = q, p

  while root:

         if root.val > q.val:

            root = root.left

        elif root.val < p.val:

            root = root.right

         else:
```

```
        return root

# Example usage:

root = TreeNode(5)

root.left = TreeNode(3, TreeNode(2), TreeNode(4))

root.right = TreeNode(6, None, TreeNode(7))

# Find the LCA of nodes with values 2 and 4

p = TreeNode(2)

q = TreeNode(4)

result = lowest_common_ancestor(root, p, q)

print(result.val)
```

OUTPUT: 3

Hard 2

You are given a string `s`. You can convert `s` to a

palindrome by adding characters in front of it.

Return *the shortest palindrome you can find by performing this transformation*.

Constraints:

- `0 <= s.length <= 5 * 104`
- `s` consists of lowercase English letters only.

CODE:

```
def shortest_palindrome(s):

    if not s:

        return s

    extended = s + "#" + s[::-1]
```

```
    lps = [0] * len(extended)
    j = 0
for i in range(1, len(extended)):
        while j > 0 and extended[i] != extended[j]:
            j = lps[j - 1]
 if extended[i] == extended[j]:
            j += 1
 lps[i] = j
    palindrome_length = lps[-1]
    return s[palindrome_length:][::-1] + s
# Example usage:
s = "abcd"
result = shortest_palindrome(s)
print(result)
```

OUTPUT: dcbabcd