Week 1 Live Coding Solution

Week 1 Live Coding Solution
- Live Coding Problem 1
  - Solution
- Live Coding Problem 2
  - Solution
- Live Coding Problem 3
  - Solution

Live Coding Problem 1

A positive integer `m` is a prime product if it can be written as `p × q`, where `p` and `q` are both primes. .

Write a Python function **prime_product(m)** that takes an integer `m` as input and returns `True` if `m` is a prime product and `False` otherwise. (If `m` is not positive, function should return `False`.)

Sample Input

```
6
```

Output

```
True
```

Solution

```python
# Solution
def factors(n):
    factorlist = []
    for i in range(1,n+1):
        if n%i == 0:
            factorlist.append(i)
    return(factorlist)
def isprime(n):
    return(factors(n) == [1,n])

def prime_product(n):
    for i in range(1,n+1):
        if n%i == 0:
            if isprime(i) and isprime(n//i):
                return(True)
    return(False)



# suffix (visible)
n = int(input())
print(prime_product(n))
```

Public Test case

Input 1

```
6
```

Output

```
True
```

Input 2

```
12
```

Output

```
False
```

Input 3

```
58
```

Output

```
True
```

Private Test case

Input 1

```
35
```

Output

```
True
```

Input 2

```
100
```

Output

```
False
```

Input 3

```
-12
```

Output

```
False
```

Input 4

```
77
```

Output

```
True
```

Live Coding Problem 2

Write a function **del_char(s,c)** that takes strings `s` and `c` as input, where `c` has length 1 (i.e., a single character), and returns the string obtained by deleting all occurrences of `c` in `s`. If `c` has length other than 1, the function should return `s`.

Sample input-1

```
banana
b
```

Output

```
anana
```

Sample input-2

```
banana
an
```

Output

```
banana
```

Solution

```python
# Solution
def del_char(s,c):
    if len(c) != 1:
        return(s)
    snew = ""
    for char in s:
        if char != c:
            snew = snew + char
    return(snew)


# Suffix (Visible)
s = input()
c = input()
print(del_char(s,c))
```

Public Test case

Input 1

```
banana
b
```

Output

```
anana
```

Input 2

```
banana
an
```

Output

```
banana
```

Input 3

```
data structure
u
```

Output

```
data strctre
```

Private Test case

Input 1

```
this is pdsa course
s
```

Output

```
thi i pda coure
```

Input 2

```
this is pdsa course
is
```

Output

```
this is pdsa course
```

Input 3

```
data structure
a
```

Output

```
dt structure
```

Input 4

```
apple
p
```

Output

```
ale
```

Live Coding Problem 3

Write a function **shuffle(l1,l2)** that takes two lists, `l1` and `l2` as input, and returns a list consisting of the first element in `l1`, then the first element in `l2`, then the second element in `l1`, then the second element in `l2`, and so on. If the two lists are not of equal length, the remaining elements of the longer list are appended at the end of the shuffled output.

Sample Input

```
[0,2,4]
[1,3,5]
```

Output

```
[0, 1, 2, 3, 4, 5]
```

Sample Input

```
[0,2,4]
[1]
```

Output

```
[0, 1, 2, 4]
```

Solution

```python
# Solution
def shuffle(l1,l2):
    if len(l1) < len(l2):
        minlength = len(l1)
    else:
        minlength = len(l2)
    shuffled = []
    for i in range(minlength):
        shuffled.append(l1[i])
        shuffled.append(l2[i])
    shuffled = shuffled + l1[minlength:] + l2[minlength:]
    return(shuffled)


# Suffix code (visible)
L1 = eval(input())
L2 = eval(input())
print(shuffle(L1,L2))
```

Public Test case

Input 1

```
[0,2,4]
[1,3,5]
```

Output

```
[0, 1, 2, 3, 4, 5]
```

Input 2

```
[0,2,4]
[1]
```

Output

```
[0, 1, 2, 4]
```

Input 3

```
[0]
[1,3,5]
```

Output

```
[0, 1, 3, 5]
```

Private Test case

Input 1

```
[1,3,5,7,9]
[2,4,6,8,10]
```

Output

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Input 2

```
[1,3,5,7,9]
[1,3,5,7,9]
```

Output

```
[1, 1, 3, 3, 5, 5, 7, 7, 9, 9]
```

Input 3

```
[1,3,5,7,9]
[2]
```

Output

```
[1, 2, 3, 5, 7, 9]
```

Input 4

```
[2]
[2,3,4,5,6,7,8,9]
```

Output

```
[2, 2, 3, 4, 5, 6, 7, 8, 9]
```