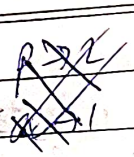
Date :.....



The question might be seeming confusing at first, but then only thing is you need to read the problem statement correctly

(s,t) are isomorphic if s becomes equal to t By the following operation

Operation:-

~~Occurrences of All character~~ Lets say you map a character in s, to a character in t. then you replace all the occurence of that character with the ~~characters in~~ new mapped character in the string t. Like this if you are able to do for all ~~the strings~~ characters and somehow (s==t) then isomorphic.

Condition. 1) one character can only be mapped to one character. one to one mapping.

Condition 2) A character already if its been mapped by another character, then another character cannot choose to be mapped by it

$x \to y$ [x is mapped y].

$p \to y$ [this cannot happen, since y is already taken].

$x \to y$, $x \to p$ (this also cannot happen)

I was overcomplicating the process, ~~whatever~~ for any character s[i] just try to find what is the corresponding t[i] and map it, if not already mapped previously.

You can keep a hashmap to store mappings and check at each stage if condition is broken

Example (egg, add)
(s t)

	(S to T mapping)	(T to S mapping)
i=0	e → a	a → e
i=1	g → d	d → g
i=2	g → d	d → g

~~e → a~~
~~g → d~~ If fine, Node breaks any Law.

Example (foo, bar)
(s t)

	(S to T mapping)	(T to S mapping)
i=0	f → b	b → f
i=1	o → a	a → o
i=2	X o → r (Fails as o.get Mapping has a value).	r → o

PAGE EDGE

Date :.....

Let's exchange the strings to show you which test case you might miss.

[foo → bar, bar → foo]

(S to T mapping)	(T to S) mapping
i=0 b → f ✓	f → b ✓
i=1 a → o ✓	o → a ✓
i=2 ✓ r → o → here we will check mappings.containsKey(r)	✗ o → r
⇓ No. Selfie.	⇓ But in this part mappings.containsKey(r)
	⇓ This is true. Hence rejected.