from functools import lru\_cache

def solve():

n = int(input().strip())

recipes = {}

for \_ in range(n):

left, right = input().strip().split("=")

potion = left.strip()

ingredients = right.strip().split("+")

recipes.setdefault(potion, []).append(ingredients)

target = input().strip()

@lru\_cache(None)

def cost(potion):

# Base case: if potion has no recipe, it's an item

if potion not in recipes:

return 0

# Try all recipes

min\_cost = float("inf")

for ing\_list in recipes[potion]:

try:

total = (len(ing\_list) - 1) + sum(cost(ing) for ing in ing\_list)

min\_cost = min(min\_cost, total)

except:

pass # skip if some ingredient cannot be brewed

return min\_cost

print(cost(target))