1. User-Friendly Password System

A website is programming an authentication system that will accept a password either if it's the correct password *or* if it's the correct password with a single character appended to it. In this challenge, your task is to implement such a system, specifically using a hashing function. Given a list of events in which either a password is set or authorization is attempted, determine if each authorization attempt will be successful or not.

The hashing function that will be used in this problem is as follows. Let f(x) be a function that takes a character and returns its decimal character code in the ASCII table. For instance f('a') = 97, f('B') = 66, and f('9') = 57. (You can find all ASCII character codes here: ASCII table.) Then, let h(s) be the hashing function that takes a string and hashes it in the following way, where $p = 131$ and $M = 10^9+7$:

$h(s) := (s[0]*P^{(n-1)} + s[1]*P^{(n-2)} + s[2]*P^{(n-3)} + ... + s[n-2]*P + s[n-1]) \mod M$

For instance, if $s = "cAr1"$, then the formula would be as follows:

$h(s) = (f('c')*131^3 + f('A')*131^2 + f('r')*131 + f('1')) \mod 10^9+7$

Test case:

Input:

2

2

setPassword cAr1

```
Authorize 1707568
Authorize 1707568
```

output:

1

1

Source code:

```
def authEvents(events):
    def f(a):
```

```
    s=0
    j=0
    n=len(a)
    for i in range(n-1,-1,-1):
        s+=ord(a[j])*(131**i)
        j+=1
    return s%((10**9)+7)
ans=[]
for i in events:
    x,y=i
    if x=='setPassword':
        p=y
    else:
        y=int(y)
        t=0
        l=0
        r=127
        if f(p)==y:
            t=1
        while l<=r:
            m=(l+r)//2
            d=f(p+chr(m))
            if d>y:
                r=m-1
            elif d<y:
                l=m+1
            else:
                t=1
                break
        if t==1:
            ans.append(1)
```

```
        else:
            ans.append(0)
    return ans
if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    events_rows = int(input().strip())
    events_columns = int(input().strip())
    events = []
    for _ in range(events_rows):
        events.append(input().rstrip().split())
    result = authEvents(events)
    fptr.write('\n'.join(map(str, result)))
    fptr.write('\n')
    fptr.close()
```

2. Generate Parentheses

Given n pairs of parentheses, write a function to *generate all combinations of well-formed parentheses*.

Example 1:

Input: n = 3
Output: ["((()))","(()())","(())()","()(())","()()()"]

Example 2:

Input: n = 1
Output: ["()"]

Source code:

```
class Solution:
    def generateParenthesis(self, n: int) -> List[str]:
        out=[]
        def f(s,l,r):
            if l==r==n:
```

```
            out.append(s)
            return
        if l>n or r>n or r>l:return
        f(s+'(',l+1,r)
        f(s+')',l,r+1)
    f('',0,0)
    return out
```