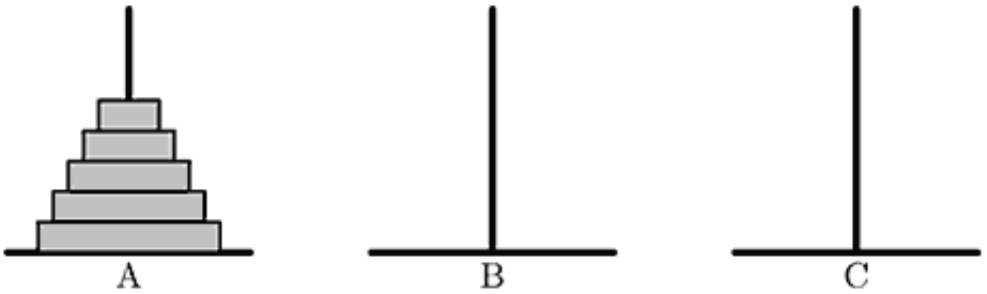
# 题意

汉诺塔由三根柱子（分别用A B C表示）和n个大小互不相同的空心盘子组成。一开始n个盘子都摞在柱子A上，
大的在下面，小的在上面，形成了一个塔状的锥形体。




对汉诺塔的一次合法的操作是指：从一根柱子的最上层拿一个盘子放到另一根柱子的最上层，同时要保证被移
动的盘子一定放在比它更大的盘子上面（如果移动到空柱子上就不需要满足这个要求）。我们可以用两个字母来描
述一次操作：第一个字母代表起始柱子，第二个字母代表目标柱子。例如，AB就是把柱子A最上面的那个盘子移到
柱子B。汉诺塔的游戏目标是将所有的盘子从柱子A移动到柱子B或柱子C上面。有一种非常简洁而经典的策略可以帮
助我们完成这个游戏。首先，在任何操作执行之前，我们以任意的次序为六种操作（AB、AC、BA、BC、CA和CB）
赋予不同的优先级，然后，我们总是选择符合以下两个条件的操作来移动盘子，直到所有的盘子都从柱子A移动到
另一根柱子：（1）这种操作是所有合法操作中优先级最高的；（2）这种操作所要移动的盘子不是上一次操作所移
动的那个盘子。可以证明，上述策略一定能完成汉诺塔游戏。现在你的任务就是假设给定了每种操作的优先级，计
算按照上述策略操作汉诺塔移动所需要的步骤数。

## 思路

设 f [k] [x] 表示把k个圆盘从柱子x移到其他柱子的最小移动数目

设 g [k] [x] 表示把k个圆盘从柱子x移到哪个柱子
由于优先级给定，那么g[1][x]已经确定了,即为优先级最高的x->y
f[1][x] = 1;

f [i] [x]:
设 y = g[i-1][x],z = 6-x-y;
1. 把i-1个圆盘从x 移到 y —— f[i-1][x]
2.把1个圆盘从x移到 z —— 1
如果 g[i-1][y] == z
3. 把 i-1个圆盘从y移到z——f[i-1][y]
综合得：f[i][x] = f[i-1][x]+1+f[i-1][y]
g[i][x] = z

如果g[i-1][y] ==x
3. 把 i-1个圆盘从y移到x —— f[i-1][y]
4. 把 1个圆盘从 z移到 y —— 1
5. 把i-1个圆盘从x 移到 y ——f[i-1][x]
综合得：f[i][x] = f[i-1][x]+1+f[i-1][y] +1 +f[i-1][x]
g[i][x] = y;

最终答案为 f[n][1]

# 代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
string cz[7];
int g[40][40];
long long f[40][40];
int n;
int main()
{
    ios::sync_with_stdio(false);
    cin>>n;
    for(int i=1;i<=6;i++)
```

```
    {
        cin>>cz[i];
    }
    for(int i=1;i<=3;i++)
    {
        for(int j=1;j<=6;j++)
        {
            if(cz[j][0]-'A'+1 == i)
            {
                f[1][i] = 1;
                g[1][i] = cz[j][1] - 'A' + 1;
                break;
            }
        }
    }
    for(int i = 2;i<=n;i++)
    {
        for(int x=1;x<=3;x++)
        {
            f[i][x] = f[i-1][x]+1;
            int y = g[i-1][x];
            int z = 6-x-y;
            if(g[i-1][y] == z)
            {
                f[i][x] += f[i-1][y];
                g[i][x] = z;
            }
            else if(g[i-1][y] == x)
            {
                f[i][x] += f[i-1][y] +1+f[i-1][x];
                g[i][x] = y;
            }
        }
    }
    cout<<f[n][1]<<endl;
}
```