FANMEETING(using Karatsuba Multiplication Algorithm)

```
# FAN MEETING
def normalize(num):    # void in C++: 리턴이 없는 함수
    """
    num의 자릿수 올림을 처리한다.
    num: vector(list)
    """
    num.append(0)     # num.push_back(0)
    # 자릿수 올림을 처리한다.
    for i in range(len(num)-1):
        if num[i] < 0:
            borrow = (abs(num[i]) + 9) // 10
            num[i+1] -= borrow
            num[i] += borrow * 10
        else:
            num[i+1] += num[i] // 10
            num[i] %= 10
    while (len(num) > 1)  & (num[-1] == 0):
        num.pop()
    return

def multiply(a, b):
    """
    a, b: vector(list)
    두 벡터 a, b를 입력받아  곱한 후 normalize한 리스트 c를 반환한다.
    즉, a * b 의 결과값을 역순으로 한 리스트를 반환한다.

    예를 들어, 123*456 = 56088 이면, a = [3,2,1]이고 b = [6, 5, 4]이며
    결과값으로 [8, 8, 0, 6, 5]를 반환한다.
    """
    # 입력된 벡터를 정수로 변환.
    c = [0] * (len(a) + len(b) + 1)
    for i in range(len(a)):
        for j in range(len(b)):
            c[i+j] += a[i] * b[j]
    normalize(c)    # 팬미팅 문제에서는 normalize하지 않음.
    return c
```

```
def addTo(a, b, k):
    """
    a += b * (10 ^ k)를 구현한다.
    a, b: vector(list)
    k: int
    """
    # a, b를 각각 정수로 변환
    a_str = str()
    b_str = str()
```

```
    for i in a:
        a_str += str(i)
    a_int = int(a_str)
    for i in b:
        b_str += str(i)
    b_int = int(b_str)
    a_ = a_int + b_int * (10 ** k)
    a = [int(i) for i in str(a_)][::-1]


def subFrom(a, b):
    """
    a -= b 를 구현한다.
    a, b = vector(list)
    """
    # a, b를 각각 정수로 변환
    a_str = str()
    b_str = str()
    for i in a:
        a_str += str(i)
    a_int = int(a_str)
    for i in b:
        b_str += str(i)
    b_int = int(b_str)
    a = [int(i) for i in str(a_int - b_int)]
```

```
def karatusba(a, b):
    """
     a, b: vector(list)
     두 벡터 a, b를 입력받아  곱한 후 normalize한 리스트 c를 반환한다.
     즉, a * b 의 결과값을 역순으로 한 리스트를 반환한다.

     예를 들어, 123*456 = 56088 이면, a = [3,2,1]이고 b = [6, 5, 4]이며
     결과값으로 [8, 8, 0, 6, 5]를 반환한다.
     """

    an, bn = len(a), len(b)

    # 기저 사례 1: a가 b보다 짧을 경우 둘을 바꾼다.
    if an < bn:
        return karatusba(b, a)
    # 기저 사례 2: a나 b가 비어있는 경우.
    if an == 0 or bn == 0:
        return [ ]
    # 기저 사례 3: a가 비교적 짧은 경우 O(N^2) 곱셈으로 변경한다.
    if an <= 50:
        return multiply(a, b)

    half = an // 2
    # a와 b를 밑에서 half와 나머지 자리로 분리한다.
    a0 = [a[0] + half] * a[0]
    a1 = [a[-1]] * (a[0] + half)
    b0 = [b[0] + min(bn, half) ] * b[0]
    b1 = [b[-1]] * (b[0] + min(bn, half))
```

```
    # z2 = a1 * b1
    z2 = karatusba(a1, b1)

    # z0 = a0 * b0
    z0 = karatusba(a0, b0)

    # a0 = a0 + a1, b0 = b0 + b1
    addTo(a0, a1, 0)
    addTo(b0, b1, 0)

    # z1 = (a0 * b0) - z0 - z2
    z1 = karatusba(a0, b0)
    subFrom(z1, z0)
    subFrom(z1, z2)

    # ret = z0 + z1 * 10^half + z2 * 10^(half * 2)
    ret = []
    addTo(ret, z0, 0)
    addTo(ret, z1, half)
    addTo(ret, z2, half + half)

    return ret
```

Key Idea

멤버	팬	결과
남자	남자	악수(1)
남자	여자	**포옹(0)**
여자	남자	**포옹(0)**
여자	여자	**포옹(0)**

```
def hugs(members, fans):
    """
    멤버와 팬의 성별이 주어질 때, 모든 멤버가 동시에 포옹하는 횟수를 출력.
    members: str, fans: str
    """
    N, M  = len(members), len(fans)
    A, B = [0] * N, [0] * M
    for i in range(N):
        if members[i] == 'M':
            A[i] = 1
        else:
            A[i] = 0
    for j in range(M):
        if fans[j] == 'M':
            B[M - j - 1] = 1
        else:
            B[M - j - 1] = 0

    # karatsuba 알고리즘에서 자리 올림은 생략한다 - multiply 함수에서 normalize 비활성화.
```

```
    K = karatusba(A, B)
    allHugs = 0
    for i in range(N-1, M):
        if K[i] == 0:
            allHugs += 1
    return allHugs
```

```
# Algospot 제출

import sys

C = int(sys.stdin.readline())
for _ in range(C):
    m = sys.stdin.readline().strip()
    f = sys.stdin.readline().strip()
    print(hugs(m, f))
```

Simpler Algorithm

```
for _ in range(int(input())):
    # 멤버와 팬의 문자열을 입력받는다.
    members, fans = input(), input()

    # 입력받은 문자 각각을 0 또는 1의 정수로 교체하고 해당 문자열을 2진수 정수로 변경한다.
    m = int(members.replace("F", "0").replace("M", "1"), 2)
    f = int(fans.replace("F", "0").replace("M", "1"), 2)

    count = 0
    for i in range(len(fans) - len(members) + 1):
    # m과 f가 같은 자리에서 동시에 1(남자, 남자)을 가지지 않는 경우. 즉 모두 포옹하는 경우.
        if int(m & f) == 0:
            count += 1
        # m의 모든 자리수를 shift한다. 즉 멤버들이 한 칸 씩 옆으로 이동한다.
        m = m << 1
    print(count)
```