```
# I may or may not have modified some of this after the competition, I do not guarantee they all
still work!
# (C) Lim Ming Yean 2018


# Q1
def reply(s):
    moves = ["8<", "O", "[]"]

    opponent, outcome = s.split()
    shift = ["D", "W", "L"].index(outcome)

    return moves[(moves.index(opponent) + shift) % 3]

def output(n, games):
    return ",".join([reply(s) for s in games])


# Q2
def look_and_say(s):
    out = ""
    last_d = s[0]
    last_pos = 0
    for i in range(1, len(s)):
        if s[i] == last_d:
            continue
        out = out + str(i - last_pos) + last_d
        last_d = s[i]
        last_pos = i
    out = out + str(len(s) - last_pos) + last_d
    return out


# Q3
def last_one_standing(N, k):
    players = list(range(1, N+1))
    cur = 0
    for i in k:
        cur = (cur + i - 1) % len(players)
        players.pop(cur)
    return players[0]


# Q4
def bacteria(k):
    if k == 1:
        return 0
    if k % 2 == 1:
        return 1 + bacteria(k + 1)
    if k % 2 == 0:
        return bacteria(k // 2)


# Q5
import string
from sympy import binomial

def trinomial(k1, k2, k3):
    return binomial(k1, k1) * binomial(k1+k2, k2) * binomial(k1+k2+k3, k3)

def analyse_pwd(pwd):
    unknown = 0
    upper = 0
    lower = 0
    numer = 0
    other = 0
```

```
    for char in pwd:
        if char == "?":
            unknown += 1
        elif char in string.ascii_lowercase:
            lower += 1
        elif char in string.ascii_uppercase:
            upper += 1
        elif char in string.digits:
            numer += 1
        else:
            other += 1

    upper_defect = max(0, 3 - upper)
    lower_defect = max(0, 3 - lower)
    numer_defect = max(0, 3 - numer)

    return unknown, upper_defect, lower_defect, numer_defect

def count_possible(pwd):
    mod = 1000000007

    x, upper_defect, lower_defect, numer_defect = analyse_pwd(pwd)

    # Inclusion-Exclusion
    s = pow(62, x, mod)

    for a in range(upper_defect):
        s -= pow(26, a, mod) * pow(36, x-a, mod) * binomial(x, a)
        s %= mod

    for b in range(lower_defect):
        s -= pow(26, b, mod) * pow(36, x-b, mod) * binomial(x, b)
        s %= mod

    for c in range(numer_defect):
        s -= pow(10, c, mod) * pow(52, x-c, mod) * binomial(x, c)
        s %= mod
        for a in range(upper_defect):
            s += pow(26, x-c, mod) * pow(10, c, mod) * trinomial(a, c, x-a-c)
            s %= mod
        for b in range(lower_defect):
            s += pow(26, x-c, mod) * pow(10, c, mod) * trinomial(b, c, x-b-c)
            s %= mod

    for a in range(upper_defect):
        for b in range(lower_defect):
            s += pow(26, a+b, mod) * pow(10, x-a-b, mod) * trinomial(a, b, x-a-b)
            s %= mod

    return s
```