```
from sys import stdin, stderr

lines = []
for line in stdin:
    lines.append(line)

REGS = {
    'A': 0,
    'B': 0,
    'C': 0,
    'D': 0,
    'E': 0,
    'F': 0,
    'P': 0
}

def opLOD(args):
    assert len(args) == 2
    X = args[0]
    V = int(args[1])

    assert X in REGS, 'Bad reg %s' % X
    REGS[X] = V

    return 1, True

def opINC(args):
    assert len(args) == 1
    X = args[0]

    assert X in REGS, 'Bad reg %s' % X
    REGS[X] += 1

    return 1, True
```

```
def opADD(args):
    assert len(args) == 2
    X = args[0]
    Y = args[1]

    assert X in REGS, 'Bad reg %s' % X
    assert Y in REGS, 'Bad reg %s' % Y
    REGS[X] += REGS[Y]

    return 1, True

def opMUL(args):
    assert len(args) == 2
    X = args[0]
    Y = args[1]

    assert X in REGS, 'Bad reg %s' % X
    assert Y in REGS, 'Bad reg %s' % Y
    REGS[X] *= REGS[Y]

    return 1, True

def opJMP(args):
    assert len(args) == 1
    V = int(args[0])

    REGS['P'] = V

    return 0, True

def opCMP(args):
    assert len(args) == 2
    X = args[0]
    Y = args[1]
```

```
    assert X in REGS, 'Bad reg %s' % X
    assert Y in REGS, 'Bad reg %s' % Y
    if REGS[X] < REGS[Y]:
        REGS['C'] = -1
    elif REGS[X] > REGS[Y]:
        REGS['C'] = 1
    else:
        REGS['C'] = 0

    return 1, True

def opJCZ(args):
    assert len(args) == 1
    V = int(args[0])

    if REGS['C'] == 0:
        REGS['P'] = V
        return 0, True

    return 1, True

def opJCP(args):
    assert len(args) == 1
    V = int(args[0])

    if REGS['C'] > 0:
        REGS['P'] = V
        return 0, True

    return 1, True

def opJCN(args):
    assert len(args) == 1
    V = int(args[0])
```

```
    if REGS['C'] < 0:
        REGS['P'] = V
        return 0, True

    return 1, True

def opHLT(args):
    assert len(args) == 0

    return 0, False

OPS = {
    'LOD': opLOD,
    'INC': opINC,
    'ADD': opADD,
    'MUL': opMUL,
    'JMP': opJMP,
    'CMP': opCMP,
    'JCZ': opJCZ,
    'JCP': opJCP,
    'JCN': opJCN,
    'HLT': opHLT
}

ROpsN = 0

while True:
    assert ROpsN < 1000, 'Too many steps'

    cur = lines[REGS['P']]
    stderr.write('S%03d: %s' % (ROpsN, cur))

    parts = cur.split()
    op = parts[0]
    args = parts[1:]
```

13. Team Slash · 2023-01-15T09:12:51Z
5/5

```
    assert op in OPS, 'Bad op %s' % op
    Pinc, cont = OPS[op](args)

    stderr.write('REGS: A=%d B=%d C=%d D=%d E=%d F=%d P=%d\n\n' % (REGS['A'], REGS[
'B'], REGS['C'], REGS['D'], REGS['E'], REGS['F'], REGS['P']))

    ROpsN += 1

    REGS['P'] += Pinc
    if not cont:
        break

print('%d %d %d %d %d %d %d' % (REGS['A'], REGS['B'], REGS['C'], REGS['D'], REGS['E
'], REGS['F'], REGS['P']))
```