```python
def exc(m,s,e):
    m = list(m)
    m[s],m[e] = m[e],m[s]
    return [''.join(m),[[e,s],ord(m[s])-96]]
def aro(pos):
    if pos in cache:
        return cache[pos]
    res = []
    dy,dx = [-1,0,1,0],[0,1,0,-1]
    y,x = divmod(pos,sx)
    for i in range(4):
        ny,nx = y + dy[i],x + dx[i]
        if -1 < ny < sy and -1 < nx < sx:
            res.append(ny * sx + nx)
    cache[pos] = res
    return res
def exp(n,m,pos=-1):
    res = []
    for i in range(size) if n > 0 else [pos]:
        if (n > 0 and m[i] != '0') or i in fix:
            continue
        for j in [j for j in aro(i) if m[j] not in ('0x' if n > 0 else 'x') and j not in fix]:
            if t == 0 and 7 in [i,j] and line not in [i,j]:
                continue
            res.append(exc(m,i,j) if n > 0 else [j,j])
    return res
def bfs(n,m,*a):
    global res
    if n == -1:
        leaf,s = a
    if n == 0:
        s,e = a
    if n == 1:
        leaf,pos,pack = a
    if n == 2:
        leaf,li,li0,n0 = a
    cur = m if n > 0 else s
    que = deque([cur])
    mkd,step = {cur:-1},{cur:-1}
    while 1:
        cur = que.popleft()
        if n == -1 and \
           leaf[cur] != '0' and cur not in hold:
            break
        if n == 0 and cur == e:
            break
        if n == 1:
            if cur == leaf or (pos != -1 and cur[pos] == pack):
                break
        if n == 2:
            li1 = ''.join([cur[i] for i in li if cur[i] != '0'])
            if li0[:n0] in li1:
                break
        for i,j in exp(n,cur if n > 0 else m,cur):
            if i not in mkd:
                que.append(i)
                mkd[i],step[i] = cur,j
    mkd[-2] = cur
    path = [step[cur]]
    while mkd[cur] != -1:
        cur = mkd[cur]
        path.append(step[cur])
    if n > 0:
        res += path[::-1][1:]
        return mkd[-2]
    return path[::-1][1:]
def sort(m,leaf,e,p=-1):
    pack = leaf[e] if p == -1 else p
    s = m.index(pack)
    r = bfs(0,m,s,e)
    for i in r:
        m = bfs(1,m,leaf,i,pack)
    fix.append(e)
    return m
def main(g_t,m,*a):
    global t,sy,sx,size,fix,res,cache
    t = g_t
    sy,sx = [[5,3],[3,4],[4,4]][t]
    size = sy * sx
    fix = []
    res = []
    cache = {}      # cache reset
    if t == 0:
        global line
        n, = a      # 라인 위치
        line = [-1,4,8,10,6][n]
        leaf = 'x0xabcdefgh0x0x'
        for i,j in [[1,'b'],[7,-1]]:
            m = sort(m,leaf,i,j)
        for i in range(1,7):
            m = bfs(2,m,leaf,[3,4,5,8,11,10,9,6],'acfhgd',i)
        fix = []
        bfs(1,m,leaf,-1,-1)
    if t == 1:
        n = m.index('x')
        leaf = list('ijklmnop0000')
        leaf.insert(n,'x')
        leaf.remove('0')
        leaf = ''.join(leaf)
        li = {4:[0,8,1], 5:[4,0,8], 6:[7,3,11], 7:[3,11,2]}
        for i in li[n]:
            m = sort(m,leaf,i)
        bfs(1,m,leaf,-1,-1)
    if t == 2:
        global hold
        leaf = ['0'] * 16
        xli = [i for i in range(16) if m[i] == 'x']
        li = list('qrstuvwyz')
        for i in [0,1,2,3,7,6,5,4,8,9,10,11,15,14,13,12]:
            if i in xli:
                leaf[i] = 'x'
            elif li:
                p = li[0]
                leaf[i] = p
                li.remove(p)
        leaf = ''.join(leaf)
        hold = {}
        pos = [i for i in [5,6,9,10] if i in xli]
        if pos:
            pos = pos[0]
            # 고정팩 위치 : [첫 번째 정렬 위치 2개, 두 번째 정렬 위치 3개]
            li = {5:[3,7,15,11,14], 6:[0,4,12,8,13],
                  9:[0,1,3,2,7], 10:[3,2,0,1,4]}[pos]
            li = [i for i in li if i not in xli]
            for i in li:
                p = leaf[i]
                if p == '0' and i not in [0,3,12,15]:
                    # 가장 가까운 정렬값이 있는 위치
                    pos0 = bfs(-1,m,leaf,i)[-1]
                    p = leaf[pos0]
                    hold[pos0] = p
                m = sort(m,leaf,i,p)
            li = {5:[1,2,6,10,9,8,4,0], 6:[2,3,7,11,10,9,5,1],
                  9:[5,6,10,14,13,12,8,4,0], 10:[6,7,11,15,14,13,9,5]}[pos]
            li0 = ''.join([leaf[i] for i in li
                           if leaf[i] != '0' and i not in hold])
            for i in range(1,len(li0)+1):
                m = bfs(2,m,leaf,li,li0,i)
            fix = []
            for i in range(16):
                if leaf[i] not in '0x' and i not in hold:
                    m = bfs(1,m,leaf,i,leaf[i])
            for i in hold:
                m = bfs(1,m,leaf,i,leaf[i])
        else:
            li = [0,1,4,3,2,7]
            for i in li:
                if leaf[i] not in '0x':
                    m = sort(m,leaf,i)
            bfs(1,m,leaf,-1,-1)
    return res

# 1. header
tl = lambda *n:tp_log(' '.join(map(str,n)))
from collections import deque
drl_report_line(OFF)
set_tool('tool wei')
set_velx(1500); set_accx(2500)
set_velj([100,150,180,225,225,225]); set_accj(400)
begin_blend(10)
ml,mj,aml,amj,tr,wt = movel,movej,amovel,amovej,trans,wait
def mjx(pos,sol=-1):
    movejx(pos,sol= sol if sol != -1 else 2)
def rml(x=0,y=0,z=0,a=0,b=0,c=0,t=0.1,vel=None,acc=None):
    aml([x,y,z,a,b,c],mod=1,v=vel,a=acc)
    mwait(0) if t == -1 else wait(t)
def rmj(x=0,y=0,z=0,a=0,b=0,c=0,t=0.1,acc=None):
    amj([x,y,z,a,b,c],mod=1,acc=acc)
    mwait(0) if t == -1 else wait(t)
def up(p,mod=0,h=-1):
    p,h = p[:],h if h != -1 else [260,260,260,400][T]
    p[2] = p[2] + h if mod else h
    return p
tool = 0
def cht(t=0.6,n = -1):
    global tool,pos
    tool = (1 - tool)
    rmj(c=-180 if tool else 180,t=t,acc=1500)
    pos = poss[tool][T]
```

```python
isgrip = False
def grip(n):
    global isgrip
    isgrip = n
    if tool:
        if not n:
            wt(0.1)
        write(40,n,b=False)
        wt(0.15)
    else:
        set_tool_digital_outputs([1,-2] if n else [-1,2])
        wt(0.25)

# 2. device communication
ser=serial_open("COM")
def ts(ad,m=[],y=0,x=0,b=True):
    if b:
        ad += 100 * (1 + T)
    k='00'+['W','R'][bool(y)]+'SB06%DW'
    k=[ord(i) for i in k]
    n=len(m) if not y else y*x
    ad = [0]*(3-len(str(ad))) + list(map(int,str(ad)))
    k+=[ord(str(abs(i))) for i in ad]
    k+=[ord(i) for i in '{:02X}'.format(n)]
    if not y:
        for i in m:
            if i<0:
                i+=2**16
            k+=[ord(j) for j in '{:04X}'.format(i)]
    ser.write([5]+k+[4])
    wait(0.02 if n == 1 else 0.05)
    k=ser.read(ser.inWaiting())
    if y:
        for i in range(0,y*x*4,4):
            v=int(k[10+i:14+i],16)
            if v&(1<<15):
                v-=2**16
            m.append(v)
        return m if len(m) > 1 else m[0]
def write(*a,b = True):
    if isinstance(a[0],int):
        a = [a]
    for i,j in a:
        ts(i,j if isinstance(j,list) else [j],b=b)
def read(ad,y=1,x=1,b=True):
    return ts(ad,[],y,x,b=b)

# - reading init
root = [-1] * 4
info = [-1] * 4
for i in range(4):
    T = i
    y,x = [5,3,4,5][i],[3,4,4,5][i]
    root[i] = read(0,y,x)[:]
info = [read(115,4,b=False).index(1)+1,-1,-1]

# - converting init
def con(t,m):
    m = ['x' if (t == 0 and i in [0,2,12,14]) else '0' if m[i] == 0 else
chr(m[i]+96) for i in range(len(m))]
```

```python
    return ''.join(m)
for i in range(3):
    root[i] = con(i,root[i])

# - get positions
def ps(pos,y,x,sy=60,sx=60):
    s = y * x
    pos = [pos] * s
    for i in range(1,s):
        if i % x:
            pos[i] = trans(pos[i-1],[-sx,0,0,0,0,0])
        else:
            pos[i] = trans(pos[i-x],[0,sy,0,0,0,0])
    return pos
elcA = ps(posx(145.05+60, 327.4, 232.01, 179.97, -90, -90.02),5,3)
elcB = ps(posx(464.77, 369.96, 233.61, 179.6, -90.01, -90),3,4)
elcC = ps(posx(-289.48, 301.08, 231.21, 179.79, 90.03, 90.04),4,4)
airA = ps(posx(145.14+60, 327.03, 139.3+60, 179.73, -90, 90),5,3)
airB = ps(posx(464.92, 369.16, 141.55+60, 179.34, -90, 90),3,4)
airC = ps(posx(-289.82, 302.39, 139.49+60, 0.16, -90.04, 90.03),4,4)
airD = ps(posx(-90.12, 404.98, 139.31+60, 0.03, -90, 90),5,5,20,20)
poss = [[elcA,elcB,elcC,None],[airA,airB,airC,airD]]

# - motions
def mt1(p,g,d=[],mod=0,h=-1,z=0,b1=True,b2=False,a1=1800,a2=500):
    tp = tr(pos[p] if type(p) == int else p,[0,0,z,0,0,0])
    ml(up(tp,mod,h),a=a1)
    if b2:
        return -1
    ml(tp,a=a2,r = 15)
    if not g:
        rml(z=-0.15,acc = 300)
    grip(g)
    if d:
        ad,val = d
        write(ad,val)
    if b1:
        ml(up(tp),r = 15)

# 5축 회전(1 : left, 0 : right)
def mt2(dire):
    global isleft
    if (isleft and dire) or (not isleft and not dire):
        return -1
    rmj(b = [180,-180][dire],acc = 1500)
    isleft = dire

# 3. Main
def RunABC():
    for i in range(len(res)):
        step,p = res[i]
        s,e = step
        a = res[i-1][1] if i else -1      # 잡았던 팩
        b = res[i+1][1] if i < len(res)-1 else -1         # 잡을 팩
        b1,b2 = p == b,p == a
        if not b2:
            mt1(s,1,[s,0],b1=False)
            rml(z=3,acc=800)
        mt1(e,0,[e,p],1,3,b2=b1)

```

```python
def RunD():
    global isleft
    m = read(0,5,5)
    line = []
    for i in range(25):
        if m[i] == 27:
            line.append(i)
    dire = abs(line[0] - line[1])
    isD = dire in [6,4]# is대각선
    isG,isS = dire == 1, dire == 5          # is가로, is세로
    if isD:           # Sequence
        li = {
        6:[1,7,13,19,2,8,14,3,9,4,5,11,17,23,10,16,22,15,21,20],
        4:[3,7,11,15,2,6,10,1,5,0,9,13,17,21,14,18,22,19,23,24]}[dire]
    else:
        n = str(int(isS)) + str([read(i) for i in range(0,25,6)].index(27) + 1)   # '가로 : 0, 세로 : 1' + '라인 위치'
        li = {'01' : [5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24],
              '02' : [0,1,2,3,4,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24],
              '03' : [5,6,7,8,9,0,1,2,3,4,15,16,17,18,19,20,21,22,23,24],
              '04' : [20,21,22,23,24,10,11,12,13,14,5,6,7,8,9,0,1,2,3,4],
              '05' : [15,16,17,18,19,10,11,12,13,14,5,6,7,8,9,0,1,2,3,4],
              '11' : [1,6,11,16,21,2,7,12,17,22,3,8,13,18,23,4,9,14,19,24],
              '12' : [0,5,10,15,20,2,7,12,17,22,3,8,13,18,23,4,9,14,19,24],
              '13' : [1,6,11,16,21,0,5,10,15,20,3,8,13,18,23,4,9,14,19,24],
              '14' : [4,9,14,19,24,2,7,12,17,22,1,6,11,16,21,0,5,10,15,20],
              '15' : [3,8,13,18,23,2,7,12,17,22,1,6,11,16,21,0,5,10,15,20]}[n]
    isleft = 0          # 5축이5 왼쪽방향인가
    dy,dx = [-1,0,1,0],[0,1,0,-1]
    for i in li + line:
        m = read(0,5,5)      # 라 파레트 정보
        pack = i + (1 if i < 23 else 2)          # 가져올 팩
        isline = i in line
        li = [[26],[7,8,9,10,11,12,19,20,21,22,23,25],[1,2,3,4,5,6,13,14,15,16,17,18]]
        z1 = [-10 * j for j in range(3) if pack in li[j]][0]       # 높이 조절1
        z2 = 60 if isline else 0      # 높이 조절2
        z = z1 + z2          # 높이 조절1 + 높이 조절2
        t = 0 if pack < 9 else 1 if pack < 17 else 2
        sy,sx = [[5,3],[3,4],[4,4]][t]
        j = read([100,200,300][t],sy,sx,False).index(pack)        # 가져올 팩이 있는 위치
        cp = tr(poss[tool][3][i],[0,0,5 + z,0,0,0])      # 팩을 밀어 넣기위해 점거할 위치
        if isline and isD:
            cp = tr(cp,[-20 if line[0] == 0 else 20,20,0,0,0,0]) # 대각선은 강제로
        else:
            for k in range(4):
                y,x = divmod(i,5)
                ny,nx = y + dy[k],x + dx[k]
                n = ny * 5 + nx
                if not (-1 < ny < 5 and -1 < nx < 5) or (not isline and m[n] == 0) or (isline and m[n] < 28):
                    cp = tr(cp,[20 * -dx[k], 20 * dy[k],0,0,0,0])      # 점거 지점 수정
        mt2(t < 2)      # 5 axis rotate
        mt1(poss[tool][t][j],1,z = z1,a1 = 3000, a2 = 2500)
        mt2(0)
        if i == line[0]:        # 라인 첫 번째는 밀어넣지 않아도 됩니다.
            mt1(i,None,z = z,b2 = True)
        else:
            mt1(cp,None,z = z,b2 = True)      # 점거 지점으로 이동
            ml(cp)
        mt1(i,0,[i,pack + [0,27][isline]],1,5,z,a1 = 250, a2 = 500)

write(30,1,b=False)
ress = []
for i in range(3):
    ress.append(main(i,root[i],info[i]))
grip(0)
mj([90,0,90,90,-90,0])
tool = 0
for i in range(4):
    if i == 2:
        mj([90,0,90,90,-90,0])
        rmj(b = 180,t =  0.5)
    T,pos = i,poss[tool][i]
    if i < 3:
        res = ress[i]
        RunABC()
    else:
        mj([90,0,90,90,90,0])
        cht()
        RunD()
write(30,1,b=False)

```