```python
def exc(m,s,e,li=-1):
    m = list(m)
    m[s],m[e] = m[e],m[s]
    step = [e,s] if li == -1 else li
    if t == 2:
        info = []
        for i in li:
            pack = 'abc'.index(i[1])+10 if i[1].isalpha() else int(i[1])
            info.append([i[0],pack])
            for j in i[0]:
                m[j] = '0'
        for i in li:
            li1,pack = i[0],i[1]
            m[li1[1]] = pack
    else:
        pack = 'abc'.index(m[s])+10 if m[s].isalpha() else int(m[s])
        info = [step,pack]
    return [''.join(m),info]
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(10 if t == 2 else size) if n > 0 else [pos]:
        if t == 2:
            ct = 1+i
            ct_ = 0
            for j in range(9-i):
                res1 = []
                for k in range(ct):
                    for l in range(0,20,10):
                        if m[j+k+l] != '0':
                            pos1 = j+k+l
                        if m[ct-k+j+l] != '0':
                            pos2 = ct-k+j+l
                    if pos1 in fix or pos2 in fix:
                        ct_ = 1
                        continue
                    pos3,pos4 = pos2,pos1
                    if (pos1 < 10 and pos2 < 10) or (pos1 > 9 and pos2 > 9):
                        pos3 = pos2+10 if pos1 < 10 else pos2-10
                        pos4 = pos1+10 if pos1 < 10 else pos1-10
                    if pos1 == pos2:
                        res1.append([[pos1,pos3],m[pos1]])
                        break
                    else:
                        res1.append([[pos1,pos3],m[pos1]])
                        res1.append([[pos2,pos4],m[pos2]])
                        if (ct == 3 and k == 1) or (ct == 5 and k == 2) or \
                           (ct == 7 and k == 3) or (ct == 9 and k == 4) or ct == 1:
                            break
```

```
                if ct_ != 0:
                    ct_ = 0
                    continue
                res.append(exc(m,-1,-1,res1))
            if i == 9:
                return res
            continue
        if (n > 0 and m[i] != '0') or i in fix:
            continue
        if t == 1 and (m[i] != '0' or (i > 2 and m[i-3] == '0')):
            continue
        if t == 1:
            li = []
            for x in range(3):
                for z in range(4):
                    pos = z * 3 + x
                    if m[pos] == '0' or (z != 3 and (pos + 3 == i or m[pos+3] !=
'0')):
                        continue
                    li.append(pos)
                    break
        for j in [j for j in (li if t == 1 else aro(i))
                  if m[j] not in ['x','0x'][n > 0] and j not in fix]:
            res.append(exc(m,i,j) if n > 0 else [j,j])
    return res
def bfs(n,m,*a):
    global res
    if n == 0:
        s,e = a
    if n == 1:
        leaf,pos,pack = a
    if n == 2:
        li,pack = a
    if n == 3:
        pack,li = a
    cur = m if n > 0 else s
    que = deque([cur])
    mkd,step = {cur:-1},{cur:-1}
    while 1:
        cur = que.popleft()
        if n == 0 and cur == e:
            break
        if n == 1:
            if (pos == -1 and cur == leaf) or \
               (pos != -1 and cur[pos] == pack):
                break
        if n == 2:
            li1 = [cur[i] for i in li]
            if pack in li1:
                break
        if n == 3:
            if ''.join([cur[i] for i in li]) == pack:
                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 main(g_t,m,*a):
    global t,sy,sx,size,fix,res,cache
    t = g_t
    sy,sx = [[4,4],[4,3],[0,0]][t]
    size = sy * sx
    fix = []
    res = []
    cache = {}      # cache reset
    if t == 0:
        pass
    if t == 1:
        leaf,pack = a
        pos = m.index(pack)
        li = [i for i in range(9)]
        li1 = [pos+i for i in range(3,12,3) if pos+i < 12]
        for i in li1[::-1]:
            if m[i] != '0':
                m = bfs(1,m,leaf,i,'0')
        m = list(m)
        m[pos] = '0'
        m = ''.join(m)
        res.append([[-1,pos], int(pack)])
        pos = leaf.index(pack)
        li1 = [pos]+[pos+i for i in range(3,12,3) if pos+i < 12]
        li = [i for i in li if i not in li1]
        for i in li:
            if leaf[i] != '0':
                pack_line = [j+i for j in range(0,12,3)]
                m = bfs(2,m,[j for j in [9,10,11] if j not in pack_line],leaf[i])
                fix.append(m.index(leaf[i]))
                m = bfs(2,m,[i],'0')
                fix.remove(m.index(leaf[i]))
                m = bfs(1,m,leaf,i,leaf[i])
                fix.append(i)
        m = bfs(2,m,[leaf.index(pack)],'0')
        m = list(m)
        m[leaf.index(pack)] = pack
        m = ''.join(m)
        res.append([[99,pos], int(pack)])
        m = bfs(1,m,leaf,-1,-1)
    if t == 2:
        leaf, = a
        for i in range(5):
            for j in range(0,20,10):
                if leaf[i+j] != '0' and leaf[i+j] != m[i+j]:
                    m = bfs(1,m,leaf,i+j,leaf[i+j])
            pos = i+j-10 if i+j > 9 else i+j+10
            fix += [i+j,pos]
        li = []
        pack = []
        for i in range(5,10):
            for j in range(0,20,10):
                if leaf[i+j] != '0':
                    li.append(i+j)
```

```
                    pack.append(leaf[i+j])
        pack = ''.join(pack)
        for i in range(1,len(li)+1):
            m = bfs(3,m,pack[:i],li[:i])
        res1 = []
        for i in res:
            res1 += i
        res = res1
    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 is -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 is -1 else wait(t)
def up(p,mod=0,h=-1):
    p,h = p[:],h if h != -1 else [-1,350,300][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]
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)
gp = 2
def gt(n,t=0.6):          # 3과제 전용
    global gp
    if n == gp:
        return -1
    write(20,n,b=False)
    write(500,n,b=False)
    wt(t *  abs(gp - n))
    gp = n

# 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 is 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)

# - reading init
root = [-1] * 3
leaf = [-1] * 3
info = [-1] * 3
for i in range(1,3):
    T = i
    y,x = [-1,4,2][i],[-1,3,10][i]
    root[i] = read(0,y,x)[:]
    leaf[i] = read(y*x+(10 if i == 2 else 0),y,x)[:]
info[1] = str(read(400,b = False))

# - converting init
def con(t,m):
    m = ['a' if i == 10 else str(i) for i in m]
    return ''.join(m)
for i in range(1,3):
    root[i] = con(i,root[i])
    leaf[i] = con(i,leaf[i])

# - get positions
def ps(pos,y,x,z,sy=40,sx=40,sz=20):
    a = y * x
    s = y * x * z
    pos = [pos] * s
    for i in range(1,s):
        if not i % a:
            pos[i] = trans(pos[i-a],[0,0,sz,0,0,0])
```

```
300         elif i % x:
301             pos[i] = trans(pos[i-1],[-sx,0,0,0,0,0])
302         else:
303             pos[i] = trans(pos[i-x],[0,sy,0,0,0,0])
304     return pos
305 elcB = posx(31.04, 529.84, 223.6, 179.97, 90.01, 90.05)
306 elcC = ps(posx(-128.82, 322.19, 228.09, 0.13, -90, -89.96),3,10,1)
307 airB = posx(-1.02, 506.67, 234.95-60, 91.94, 90, -90)
308 airC = ps(posx(-130.62, 323.24, 136.35+60, 0.37, -90.02, 90.05),3,10,1)
309 poss = [[None,elcB,elcC],[None,airB,airC]]
310
311 # - motions
312 def mt1(p,g,d=[],mod=0,h=-1,z=0,b1=True,b2=False,a1=1800,a2=1000):
313     tp = tr(pos[p] if type(p) == int else p,[0,0,z,0,0,0])
314     ml(up(tp,mod,h),a=a1)
315     if b2:
316         return -1
317     ml(tp,a=a2,r = 15)
318     if not g:
319         rml(z=-0.15,acc = 300)
320     grip(g)
321     if d:
322         ad,val = d
323         write(ad,val)
324     if b1:
325         ml(up(tp),r = 15)
326
327 # 3. Main
328 def RunB():
329     now = 2
330     for i in range(len(res)):
331         step,p = res[i]
332         s,e = step
333         isAir = s in [-1,99]
334         getxz = lambda n: ((n % 3) + 1, 20 * (n // 3))
335         if isAir:
336             pos,n = e, s == -1
337             x,z = getxz(e)
338             cht()
339             gt(x)
340             mt1(airB,n,[pos,[p,0][n]],z = z)
341             cht()
342             ml(up(elcB))
343         else:
344             for j in range(2):
345                 pos = step[j]
346                 x,z = getxz(pos)
347                 gt(x)
348                 mt1(elcB,not j,[pos,[0,p][j]],z = z)
349
350 def RunC():
351     i = 0
352     while i < len(res):
353         step,p = res[i]
354         areaLen = abs((step[0] % 10) - (step[1] % 10)) + 1
355         li = res[i:i+areaLen]       # 내림차춘으로 변경
356         li.sort(key = lambda n : n[0][0] % 10)
357         C3 = []
358         for j in range(2):
359             cht()
```

```
            for n in range(areaLen):
                step,p = li[n]
                s,e = step
                if not j and s < 10:        # C1 -> C3
                    z = -(20 if p < 4 else 10 if p < 9 else 0)
                    mt1(s,1,[s,0],z=z)
                    mt1(e+10,0,[e+10,p],z=z)
                    C3.append([e+10,p])
                if j and p not in [k[1] for k in C3]:       # C2 -> C1
                    mt1(s,1,[s,0])
                    mt1(e,0,[e,p])
        C3.sort(key = lambda n : n[0] % 10)       # 내림차춘으로 변경
        for j in range(len(C3)):        # C3 -> C2
            s,p = C3[j]
            e = s - 10
            mt1(s,1,[s,0])
            mt1(e,0,[e,p])
        i += areaLen


write(30,1,b=False)
ress = [-1]
for i in range(1,3):
    if info[i] != -1:
        ress.append(main(i,root[i],leaf[i],info[i]))
    else:
        ress.append(main(i,root[i],leaf[i]))
grip(0)
mj([90,0,90,90,90,0])
tool = 0
for i in range(1,3):
    T,pos,res = i,poss[tool][i],ress[i]
    Run = [-1,RunB,RunC][i]
    Run()
write(30,1,b=False)

```