In [21]:

```
from copy import *

matrizInicial = [
    [0,0,0,0,0],
    [0,0,1,0,0],
    [0,1,2,1,0],
    [0,0,5,0,0],
    [0,0,0,0,0],
]

class State:
    def __init__(self, pai, matriz):
        self.pai = pai
        self.matriz = deepcopy(matriz)

def initialState():
    return State(None, matrizInicial)

def showState(s):
    for linha in s.matriz:
        print(linha)
    print("")

def goal(s):
    for linha in s.matriz:
        for elemento in linha:
            if(elemento != 0):
                return False

    return True

def mCima(s, linha, coluna):
    origem = s.matriz[linha][coluna]
    destino = s.matriz[linha - 1][coluna]
    if (origem == 0 or destino == 0):
        return None
    ret = State(s, s.matriz)
    if(origem == destino):
        ret.matriz[linha][coluna] = 0
        ret.matriz[linha - 1][coluna] = 0
    else:
        ret.matriz[linha][coluna] = 0
        ret.matriz[linha - 1][coluna] = origem + destino
    return ret

def mBaixo(s, linha, coluna):
    origem = s.matriz[linha][coluna]
    destino = s.matriz[linha + 1][coluna]
    if (origem == 0 or destino == 0):
        return None
    ret = State(s, s.matriz)
    if(origem == destino):
        ret.matriz[linha][coluna] = 0
        ret.matriz[linha + 1][coluna] = 0
    else:
        ret.matriz[linha][coluna] = 0
        ret.matriz[linha + 1][coluna] = origem + destino
        return ret

def mEsquerda(s, linha, coluna):
    origem = s.matriz[linha][coluna]
    destino = s.matriz[linha][coluna - 1]
    if (origem == 0 or destino == 0):
        return None
    ret = State(s, s.matriz)
    if(origem == destino):
```

```
        ret.matriz[linha][coluna] = 0
        ret.matriz[linha][coluna - 1] = 0
    else:
        ret.matriz[linha][coluna] = 0
        ret.matriz[linha][coluna - 1] = origem + destino
        return ret

def mDireita(s, linha, coluna):
    origem = s.matriz[linha][coluna]
    destino = s.matriz[linha][coluna + 1]
    if (origem == 0 or destino == 0):
        return None
    ret = State(s, s.matriz)
    if(origem == destino):
        ret.matriz[linha][coluna] = 0
        ret.matriz[linha][coluna + 1] = 0
    else:
        ret.matriz[linha][coluna] = 0
        ret.matriz[linha][coluna + 1] = origem + destino
        return ret

def expand(s):
    ret = []
    for i in range(1, len(s.matriz) - 1):
        for j in range(1, len(s.matriz[i]) - 1):
            filho = mCima(s, i, j)
            if (filho != None):
                ret.append(filho)
            filho = mBaixo(s, i, j)
            if (filho != None):
                ret.append(filho)
            filho = mDireita(s, i, j)
            if (filho != None):
                ret.append(filho)
            filho = mEsquerda(s, i, j)
            if (filho != None):
                ret.append(filho)
    return ret

def showPath(s):
    if(s == None):
        return
    showPath(s.pai)
    showState(s)

queue = []

def enqueue(s):
    queue.append(s)

def dequeue():
    return queue.pop(0)

s = initialState()
enqueue(s)

while(queue):
    s = dequeue()

    if(goal(s)):
        showPath(s)
        break

    children = expand(s)
    for child in children:
        enqueue(child)
```