a)

MinCoinsTuple(a, c_0, . . . , c_{k-1})

```
1       S ← new array [0, . . . , a]
2       M ← new array [0, . . . , a]
3       if a == 0 then return [0]*k
4       foreach i from 0 to a do
5               if i = 0
6                       then M[i] ← 0
7                       S[0] = [0]*k
8               else
9                       M[i] ←∞
10                      foreach j from 0 to k − 1 do
11                              if i ⩾ cj then
12                                      if M[i] > 1 +M[i − cj]
13                                              then
14                                                      temp = S[i-cj]
15                                                      M[i] ← 1 +M[i − cj]
16                                                      temp[j] += 1
17                      S[i] = temp
```

b)

```
def MinCoinsTuple(a,c):
    S=[0]*k
    if a == 0 then return S
    minCoins = MinTotalCoins(a,c)
    foreach j from 0 to k-1 do:
        denom = c[j]
        lower = 0
        upper = ⌊a/denom⌋
        if upper > 0 then d = BinSearchCoin(a,denom,lower,upper+1,minCoins,c)
        else
            d = 0
        S[j] = d
    return S


def BinSearchCoin(a,denom,lower,upper,minCoins,c):
    d = 0
    while lower <= upper:
        mid = ⌈(lower+upper)/2⌉
        if MinTotalCoins(a-denom*mid, c) == (minCoins-mid):
            d = mid
            return d
        elif MinTotalCoins(a-denom*mid, c) < (minCoins-mid):
            lower = mid + 1
        else
            upper = mid – 1
    return d
```

c)

$$M[x, n_0, \dots, n_{k-1}]$$

$$= \begin{cases} \infty & if\ x < 0\ or\ \underset{i \in \{0,..k-1\}}{n_i < 0} \\ 0 & if\ x = 0 \\ \min\limits_{i \in \{0,\dots k-1\}} \{M[x - c_i, n_0, \dots, n_i - 1, \dots n_{k-1}]\} & otherwise \end{cases}$$