3. Dar una definición de la función cambio utilizando la técnica de programación dinámica a partir de cada una de las siguientes definiciones recursivas (backtracking):

(a)

$$cambio(i,j) = \begin{cases} 0 & j = 0 \\ 1 + \min_{i' \in \{1,2,\ldots,i \mid d_{i'} \leq j\}} (cambio(i', j - d_{i'})) & j > 0 \end{cases}$$

```
fun cambio1(d: array[1..n] of nat, monto: nat) ret cantidad: nat
    var tabla: array[0..n,0..monto] of nat   {- tabla[i,j] = cambio1(i,j) -}

    {- Caso 1 -}
    for i := 0 to n do
        tabla[i,0] := 0
    od

    {- Caso 2 -}
    for i := 0 to n do
        for j := 1 to monto do
            minimo := ∞
            for i' := 1 to i do
                if d[i'] ≤ monto then
                    minimo := minimo `min` tabla[i',j-d[i']]
                fi
            od
            tabla[i,j] := 1 + minimo
        od
    od

    cantidad := tabla[n,monto]
end fun
```

(b)

$$cambio(i,j) = \begin{cases} 0 & j = 0 \\ \infty & j > 0 \wedge i = n \\ cambio(i+1,j) & d_i > j > 0 \wedge i < n \\ \min(cambio(i+1,j), 1 + cambio(i, j - d_i)) & j \geq d_i > 0 \wedge i < n \end{cases}$$

```
fun cambio2(d: array[1..n] of nat, monto: nat) ret cantidad: nat
    var tabla: array[0..n,0..monto] of nat   {- tabla[i,j] = cambio2(i,j) -}

    {- Caso 1 -}
    for i := 0 to n do
        tabla[i,0] := 0
    od

    {- Caso 2 -}
    for j := 1 to monto do
        tabla[n,j] := ∞
    od

    for i := n-1 downto 1 do
        for j := 1 to monto do
            if d[i] > j then   {- Caso 3 -}
                tabla[i,j] := tabla[i+1,j]
            else   {- Caso 4 -}
                tabla[i,j] := tabla[i+1,j] `min` 1 + tabla[i,j-d[i]]
```

```
            fi
        od
    od

    cantidad := tabla[n,monto]
end fun
```