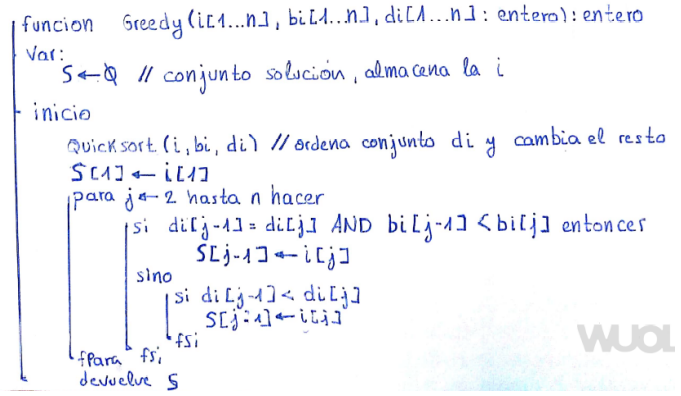
```
funcion Devolver-cambio (int P): conjunto de monedas(X)
  const C={1,2,5,10,20,50,100,200} // C=monedas disponibles; conj. candidatos
  int X[1..N] ;                    // X= conjunto que contendrá la solución
  actual = 0                       //suma acumulada de la cantidad procesada
  para i = 1 hasta N
     X[i] = 0                      // inicialización(X)
  fpara
  mientras actual ≠ P              // no solución(X)
    j = el mayor elemento de C tal que C[j] ≤ (P-actual) // seleccionar(C)
    si j=0 entonces         // Si no existe ese elemento => no factible(j)
       devolver "No existe solución";
    fsi
    X[j] = (P-actual) div C[j]     // insertar(C,X)
    actual = actual + C[j]*X[j]
  fmientras
  devolver X                       // objetivo(X)
ffuncion
```

```
funcion mochila(P(1...N), B(1...N) : entero, M : entero)
    S(1...N) ← (0, ... 0) // Conjunto solución
    pesoAct ← 0
    ordenados(1...N) ← ordenar(P, B) // Ordena los artículos en función del
    beneficio/peso, devolviendo un array con los índices de las posiciones
    i ← 1
    mientras (i <= N Y pesoAct <= M) hacer
        pos ← ordenados(i) // articulo con mayor b/p en cada iteración
        peso ← P(pos)
        si (pesoAct + peso <= M) entonces
            S(pos) ← 1
            pesoAct ← pesoAct + peso
        fsi
        i ← i + 1
    fmientras
    devuelve S
ffuncion
```




```
funcion selectorAct(C(1...N), F(1...N) : entero)
    S(1...N) ← {1} // Conjunto solución (N es el número máximo posible)
    // C(1...N) son los instantes de comienzo de la actividad i-esima
    // F(1...N) son los instantes de finalización de la actividad i-esima
    // Tanto C como F están ordenados según el criterio: (i<j ⇒ f(i)≤f(j))
    z ← 1 // Última actividad seleccionada, inicialmente es la 1
    para i = 2 hasta n (inc 1) hacer
        si (C(i) >= F(z)) entonces
            S ← S U {i}
            z ← i
        fsi
    fpara
    devuelve S
ffuncion
```