- Describa cuál es el criterio de selección.
- ¿En qué estructuras de datos representará la información del problema?
- Explique el algoritmo, es decir, los pasos a seguir para obtener el resultado. No se pide que "lea" el algoritmo ("se define una variable x", "se declara un for"), si no que lo explique ("se recorre la lista/el arreglo/" o "se elije de tal conjunto el que satisface...").
- Escriba el algoritmo en el lenguaje de programación de la materia.

4. En numerosas oportunidades se ha observado que cientos de ballenas nadan juntas hacia la costa y quedan varadas en la playa sin poder moverse. Algunos sostienen que se debe a una pérdida de orientación posiblemente causada por la contaminación sonora de los océanos que interferiría con su capacidad de inter-comunicación. En estos casos los equipos de rescate realizan enormes esfuerzos para regresarlas al interior del mar y salvar sus vidas.

   Se encuentran n ballenas varadas en una playa y se conocen los tiempos $s_1, s_2, \ldots, s_n$ que cada ballena es capaz de sobrevivir hasta que la asista un equipo de rescate. Dar un algoritmo voraz que determine el orden en que deben ser rescatadas para salvar el mayor número posible de ellas, asumiendo que llevar una ballena mar adentro toma tiempo constante t, que hay un único equipo de rescate y que una ballena no muere mientras está siendo regresada mar adentro.

   - **Criterio de selección**
     Elige la ballena con menor tiempo de supervivencia.

   - **Estructuras de datos**
     Planteo a las ballenas como una tupla de dos elementos con un identificador del animal y el tiempo de vida que le queda.

```
type Ballena = tuple
                 id: nat
                 tiempo_de_vida: nat
               end tuple

fun salvar_ballenas(t: nat, B: Set of Ballena) ret res: List of Ballena
```

     Las ballenas varadas se acomodarán en un conjunto y luego aquellas que pueden ser salvadas, derivadas a una lista.

   - **Descripción de cómo se soluciona el problema**
     Primero salva la ballena más pronta a morir y luego la descarta junto a aquellas que murieron mientras la estaba rescatando. Se repite hasta que al final no queden ballenas por salvar.

   - **Definición del algoritmo**

```
type Ballena = tuple
                 id: nat
                 tiempo_de_vida: nat
               end tuple

fun salvar_ballenas(t: nat, B: Set of Ballena) ret res: List of Ballena
    var B_aux: Set of Ballena
    var b: Ballena
    var hora: nat

    hora := 0
    B_aux := copy_set(B)
    res := empty_list()

    while not is_empty_set(B_aux) do
        b := elegir_ballena(B_aux)
```

```
        addr(res,b)
        elim(B_aux,b)
        hora := hora + t
        descartar_ballenas_muertas(B_aux,hora)
    od

    destroy_set(B_aux)
end fun

fun elegir_ballena(B: Set of Ballena) ret res: Ballena
    var B_aux: Set of Ballena
    var b: Ballena
    var min_tiempo_restante: nat

    B_aux := copy_set(B)
    min_tiempo_restante := ∞

    while not is_empty_set(B_aux) do
        b := get(B_aux)
        if b.tiempo_de_vida < min_tiempo_restante then
            min_tiempo_restante := b.tiempo_de_vida
            res := b
        fi
        elim(B_aux,b)
    od

    destroy_set(B_aux)
end fun

proc descartas_ballenas_muertas(in/out B: Set of Ballena, in h: nat)
    var B_aux: Set of Ballena
    var b: Ballena

    B_aux := copy_set(B)

    while not is_empty_set(B_aux) do
        b := get(B_aux)
        if b.tiempo_de_vida < h then
            elim(B,b)
        fi
        elim(B_aux,b)
    od

    destroy_set(B_aux)
end proc
```