- 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.

5. Sos el flamante dueño de un teléfono satelital, y se lo ofrecés a tus n amigos para que lo lleven con ellos cuando salgan de vacaciones el próximo verano. Lamentablemente cada uno de ellos irá a un lugar diferente y en algunos casos, los períodos de viaje se superponen. Por lo tanto es imposible prestarle el teléfono a todos, pero quisieras prestárselo al mayor número de amigos posible.

   Suponiendo que conoces los días de partida y regreso (p_i y r_i respectivamente) de cada uno de tus amigos, ¿cuál es el criterio para determinar, en un momento dado, a quien conviene prestarle el equipo?

   Tener en cuenta que cuando alguien lo devuelve, recién a partir del día siguiente puede usarlo otro. Escribir un algoritmo voraz que solucione el problema.

   - **Criterio de selección**
     Elige al amigo que tenga el menor día de regreso.

   - **Estructuras de datos**
     Planteo a los amigos como una tupla de tres elementos con el nombre del amigo, su día de partida y día de regreso.

```
type Amigos = tuple
                id: string
                partida: nat
                regreso: nat
              end tuple

fun prestar_telefono(A: Set of Amigos) ret res: List of Amigos
```

     Los amigos que están esperando el teléfono están ordenados en un conjunto, mientras que luego los que usaron el teléfono se derivan a una lista.

   - **Descripción de cómo se soluciona el problema**
     Escoge el amigo con el menor tiempo de regreso y lo descarta de la lista junto a aquellos que todavía están en viaje y no volvieron. Se repite hasta que no queden más amigos con viajes programados.

   - **Definición del algoritmo**

```
type Amigos = tuple
                id: string
                partida: nat
                regreso: nat
              end tuple

fun prestar_telefono(A: Set of Amigos) ret res: List of Amigos
    var A_aux: Set of Amigos
    var a: Amigos
    var dia_de_regreso: nat

    A_aux := copy_set(A)
    res := empty_list()

    while not is_empty_set(A_aux) do
        a := elegir_amigo(A_aux)
```

```
        dia_de_regreso := a.regreso
        addr(res,a)
        elim(A_aux,a)
        descartar_amigos_en_viaje(A_aux,dia_de_regreso)
    od

    destroy_set(A_aux)
end fun

fun elegir_amigo(A: Set of Amigos) ret res: Amigos
    var A_aux: Set of Amigos
    var a: Amigos
    var menor_dia_de_regreso: nat

    A_aux := copy_set(A)
    menor_dia_de_regreso := ∞

    while not is_empty_set(A_aux) do
        a := get(A_aux)
        if a.regreso < menor_dia_de_regreso then
            menor_dia_de_regreso := a.regreso
            res := a
        fi
        elim(A_aux,a)
    od

    destroy_set(A_aux)
end fun

proc descartar_amigos_en_viaje(in/out A: Set of Amigos, in d: nat)
    var A_aux: Set of Amigos
    var a: Amigos

    A_aux := copy_set(A)

    while not is_empty_set(A_aux) do
        a := get(A_aux)
        if a.partida ≤ d then
            elim(A,a)
        fi
        elim(A_aux,a)
    od

    destroy_set(A_aux)
end proc
```