Contents

3)

```
type localidad = tuple
  name: String,
  dist: Nat
end tuple

fun cargarCombustible(A: Nat, l1: List of localidad) ret res : List of localidad
  var l2: list of localidad
  var distSum: nat
  distSum:= 0
  res:= empty_list()
  l2:= copy_list(l1)
  while !list_is_empty(l2) do
    head:= list_head(l2)
    list_tail(l2)
    if distSum + head.dist ≤ A then
      distSum:= distSum + head.dist
      prevHead:= head
    else
      list_addr(res, prevHead)
      distSum:= head.dist
    fi
  od
end fun
```

4)

```
type ballena = tuple
  id: nat
  timeLeft: nat
end tuple

fun salvarBallenas(ballenas: set of ballena ) ret res : Queue of Ballena
  var ballenasVivas: set of ballena
  var tiempo: nat
  ballenasVivas:= set_copy(ballenas)
  res:= empty_queue()
  while !set_is_empty(ballenasVivas) do
    salvada:= seleccionarBallena(ballenasVivas)
    enqueue(res, salvada)
    set_elim(ballenasVivas, salvada)
    tiempo:= tiempo + t
    ballenasVivas:= quitarMuertas(ballenasVivas, tiempo)
  od
end fun
```

```
fun seleccionarBallena(ballenas: set of ballena) ret res : ballena
  var ballenas2: set of ballena
  var head: ballena
  res:= set_get(ballenas2)
  while !set_is_empty(ballenas2) do
    head:= set_get(ballenas2)
    set_elim(ballenas, head)
    if head.timeLeft < res.timeLeft then
      res:= head
    fi
  od
end fun
```

```
proc eliminarBallenas(in/out ballenas: set of ballena, in tiempo: nat)
  var head: ballena
  var ballenas2: set of ballena
  ballenas2:= set_copy(ballenas)
  while !set_is_empty(ballenas2) do
    head:= set_get(ballenas2)
    set_elim(ballenas2, head)
    if head.timeLeft < tiempo then
      set_elim(ballenas, head)
    fi
  od
end fun
```

5)

```
type amigo = tuple
  partida: nat,
  regreso: nat
end tuple
```

```
fun aQuienPrestar(amigos: set of amigo) ret res : Queue of Amigo
  var amigosPendientes: set of amigo
  var seleccion: amigo
  var dia: nat
  res:= empty_queue()
  amigosPendientes:= set_copy(amigos)
  while !set_is_empty(amigosPendientes) do
    seleccion:= seleccionarAmigo(amigosPendientes)
    set_elim(amigosPendientes, seleccion)
    dia:= seleccion.regreso
    enqueue(res, seleccion)
    eliminarAmigos(amigosPendientes, dia)
  od
end fun
```

```
fun seleccionarAmigo(amigos: set of amigo) ret res : amigo
  var amigos2: set of amigo
  var head: amigo
  res:= set_get(amigos2)
  while !set_is_empty(amigos2) do
    head:= set_get(amigos2)
    set_elim(amigos, head)
    if head.regreso < res.regreso then
      res:= head
    fi
  od
end fun
```

```
proc eliminarAmigos(in/out amigos: set of amigo, in dia: nat)
  var head: amigo
  var amigos2: set of amigo
  amigos2:= set_copy(amigos)
  while !set_is_empty(amigos2) do
    head:= set_get(amigos2)
    set_elim(amigos2, head)
    if head.partida ≤ dia then
      set_elim(amigos, head)
    fi
  od
end fun
```

6)

```
type factura = tuple
  tmin: nat,
  tmax: nat
end tuple
```

```
fun cuandoSacar(facturas: set of factura) ret res : Queue of nat
  var facturasPendientes: set of factura
  var seleccion: factura
  var t: nat
  res:= empty_queue()
  facturasPendientes:= set_copy(facturas)
  while !set_is_empty(facturasPendientes) do
    seleccion:= seleccionarFactura(facturasPendientes)
    set_elim(facturasPendientes, seleccion)
    t:= seleccion.tmax
    enqueue(res, t)
    eliminarFactura(facturasPendientes, t)
  od
end fun
```

```
fun seleccionarFactura(facturas: set of factura) ret res : factura
  var facturas2: set of factura
  var head: factura
  res:= set_get(facturas2)
  while !set_is_empty(facturas2) do
    head:= set_get(facturas2)
    set_elim(facturas, head)
    if head.tmax < res.tmax then
      res:= head
    fi
  od
end fun
```

```
proc eliminarFacturas(in/out facturas: set of factura, in t: nat)
  var head: factura
  var facturas2: set of factura
  facturas2:= set_copy(facturas)
  while !set_is_empty(facturas2) do
    head:= set_get(facturas2)
    set_elim(facturas2, head)
    if head.tmin ≤ t then
      set_elim(facturas, head)
    fi
  od
end fun
```

7)

a)

```
type tripulante = tuple
  id: nat
  oxPerMin: nat
end tuple
```

```
fun salvarTripulantes(tripulantes: set of tripulante ) ret res : Queue of Ballena
  var tripulantesVivos: set of tripulante
  var oxigenoActual: int
  oxigenoActual:= C
  tripulantesVivos:= set_copy(tripulantes)
  res:= empty_queue()
  while !set_is_empty(tripulantesVivos) do
    salvado:= seleccionarTripulante(tripulantesVivos)
    enqueue(res, salvado)
    set_elim(tripulantesVivos, salvado)
    oxigenoActual:= oxigenoActual - oxigenoTotalPorMin(tripulantesVivos) * t
    if oxigenoActual ≤ 0 then
      set_destroy(tripulantesVivos)
      tripulantesVivos:= empty_set()
    fi
  od
end fun

fun seleccionarTripulante(tripulantes: set of tripulante) ret res : tripulante
  var tripulantes2: set of tripulante
  var head: tripulante
  res:= set_get(tripulantes2)
  while !set_is_empty(tripulantes2) do
    head:= set_get(tripulantes2)
    set_elim(tripulantes, head)
    if head.oxPerMin ≥ res.oxPerMin then
      res:= head
    fi
  od
end fun

fun oxigenoTotalPorMin(tripulantes: set of tripulante ) ret res : nat
  var tripulantes2: set of tripulante
  var head: tripulante
  tripulantes2:= set_copy(tripulantes)
  res:= 0
  while !set_is_empty(tripulantes2) do
    head:= set_get(tripulantes2)
    set_elim(tripulantes, head)
    res:= res + head.oxPerMin
  od
end fun
```

8)

```
type tronco = tuple
  temp: nat
  duracion: nat
end tuple

fun quemarTroncos(troncos: set of tronco ) ret res : Queue of tronco
  var troncosVivos: set of tronco
  var tiempo: nat
  tiempo:= 0
  troncosVivos:= set_copy(troncos)
  res:= empty_queue()
  while !set_is_empty(troncosVivos)∧time≤840 do
    if tiempo ≤ 480 then
      quemado:= seleccionarTronco(troncosVivos, K1)
    else
      quemado:= seleccionarTronco(troncosVivos, K2)
    fi
    set_elim(troncosVivos, quemado)
    enqueue(res, quemado)
    tiempo:= tiempo + quemado.duracion
  od
end fun

fun seleccionarTronco(troncos: set of tronco, tempMin: nat) ret res : tronco
  var troncos2: set of tronco
  var head: tronco
  res:= set_get(troncos2)
  while !set_is_empty(troncos2) do
    head:= set_get(troncos2)
    set_elim(troncos, head)
    if head.duracion ≥ res.duracion ∧ head.temp ≥ tempMin then
      res:= head
    fi
  od
end fun
```

9)

```
type bar = tuple
  happyh: nat,
  price: float
end tuple
```

```
fun sobredosisDeLimonada(bares: set of bar) ret res : float
  var baresPendientes: set of bar
  var seleccion: bar
  var t: nat
  t:= 0
  res:= empty_queue()
  baresPendientes:= set_copy(bares)
  while !set_is_empty(baresPendientes) ∧ t ≤ 8 do
    seleccion:= seleccionarBar(baresPendientes)
    set_elim(baresPendientes, seleccion)
    if seleccion.happyh ≤ t then
      res:= res + seleccion.price
    else
      res:= res + seleccion.price * 2
    fi
    t:= t + 1
  od
end fun

fun seleccionarBar(bares: set of bar) ret res : bar
  var bares2: set of bar
  var head: bar
  res:= set_get(bares2)
  while !set_is_empty(bares2) do
    head:= set_get(bares2)
    set_elim(bares, head)
    if head.happyh ≤ res.happyh then
      if head.happyh = res.happyh ∧ head.price ≥ res.price then
        skip
      else
        res:= head
      fi
    fi
  od
end fun
```