En el modelo

1) Tipo [(Indice,Dato)] representa un mensaje

Type Dato = <completar>

Type Indice = <completar>

Type Paquete = <completar>

<completar> Mensaje = [Paquete]

a) UltimPaq toma f y Mensaje xs devuelve Paquete correspondiente en el primero o último lugar en el orden del mensaje comp. dependiendo que funcion f se utilice

Ej: ultimoPaq ascend [(3,Mundo),(1,Hola),(4,!), (2, " ")] = (4, "! ")

UltPaq f < completar > = < completar >

UltPaq f (x : y : xs) | f <completar > <completar > = < completar >

| f otherwise < completar > = < completar >

ascendente < completar > < completar > = < completar >

c) Faltantes, solo con listas por comprensión, dado Mensaje xs ordenado ascendentemente devuelve [Indice] de los paquetes faltantes.

Faltantes [(1,"Hola"), (3,"Mundo"), (6,"!"] = [2,4,5]. Puede usar concat

b) En orden Asc usando foldr y map, dado un mensaje retorna si los paquetes estan ordenados ascendentemente o no.

```
EnOrdenAsc xs  = let as = map  < completar >  xs  (retorna Indice)
                     ady = zip os (tail os)  (completar el comentario diciendo que hace)
                     in foldr < completar > True ady
```

2) Data BST a = E | N (BST a) a (BST a)

Type msnTree = BST (Indice,Dato)

a) Suponga que los paquetes llegan (1,"Hola"), (s,"1"),(6,"!"), (3,"Mundo") ,(2," ") ,(4," ") Escribir la representacion del mensaje en el tipo msnTree.

b) Definir mensaje msnTree , retorna Info completa que resulta de pegar datos en los paquetes en orden. Ej: mensaje[(1,"Hola"),(3,"Mundo"),(2," ")] = "Hola mundo"

3) Dar el tipo de ruda f g h = f(gh)(gh)

4) Para el tipo leftist heaps:

a – Que es una espina en un arbol? Que es el rango de una leftist Heaps?

b – Invariante de Heaps y leftist Heaps

c – Consecuencias que tienen el/los invariantes de leftist Heaps