MOISE GABRIEL

FUNCTIONAL PROGRAMMING - 1997

QUESTION 2

Note: This question is very similar to Q3 from 2012, which was set as homework for the Christmas holiday.

(a)

```
> merge :: [a] -> [a] -> [a]
> merge [] ys = ys
> merge (x:xs) ys = x : merge ys xs
```

This function alternates the elements of the two lists until one of them becomes empty, or it goes forever if both lists are infinite.

(b)

For allpairs xs ys to contain all pairs that can be formed with one element from xs and the other one from ys, we need to create a list in which we can find any given pair with this property in a finite number of steps.

Therefore, we will create a function that, given an unknown number of lists (finite / infinite), it returns a list that contains all the elements that appear in those lists.

```
> mergeLists :: [[a]] -> [a]
> mergeLists = foldr merge []
```

We used merge from (a) because this function, given two lists, returns a lists where all the elements appear.

Now, we will use this function on the list of lists each containing one element of xs, and all the elements of ys:

```
> allpairs :: [a] -> [b] -> [(a,b)]
> allpairs xs ys = mergeLists [ [ (x,y) | y <- ys ] | x <- xs ]
```

```
allpairs [1,2] [1,2] = [(1,1), (2,1), (1,2), (2,2)]
take 6 ( allpairs [1,2] [1..] ) = [(1,1), (2,1), (1,2), (2,2), (1,3), (2,3)]
take 6 ( allpairs [1..] [1,2] ) = [(1,1), (2,1), (1,2), (3,1), (2,2), (4,1)]
take 6 ( allpairs [1..] [1..] ) = [(1,1), (2,1), (1,2), (3,1), (1,3), (2,2)]
```

(c)

```
> data Btree = Nil | Node Btree Btree

> allTrees :: [Btree]
> allTrees = Nil : rest
>     where rest = map (uncurry Node) (allpairs allTrees allTrees)
```

This way, by lazy evaluation, we first create the Btrees with few nodes, and from them we make all the pairs which get applied (uncurry Node), which makes new Btrees with more nodes and so on.

take 6 allTrees = [Nil, Node Nil Nil, Node (Node Nil Nil) Nil, Node Nil (Node Nil Nil), Node (Node (Node Nil Nil) Nil) Nil, Node Nil (Node (Node Nil Nil) Nil)]