CMSC 331: Principles of Programming Language
Homework 2

1. Write a function called "concatN", with the following type spec:

   concatN: int -> string-> string

   here is a couple of sample runs:

   ```
   - concatN 0 "x"
   val it = ""  : string
   - concatN 4 "x"
   val it = "xxxx" : string
   ```

2. a) Write a function called "concat", with the following type spec:

   concat: string list -> string

   here is a couple of sample runs:

   ```
   - concat ["x", "y", "z"];
   val it = "xyz": string
   ```

   Then, given the functions:

   ```
   fun curry f x y = f (x,y);
   fun uncurry f (x,y) = f x y;
   ```

   Determine the type specs of:

   b) fun smush (a,b) = concat [a,b];

   c) val smush2 = curry smush;

3. Given the following data type:

   ```
   datatype cTree = EMPTY |
                NODE of string * int * cTree * cTree;
   ```

   (Where the node basically holds a string, and a count of how many times the string occurs)

   you may presume that the append function is available (as defined in class).

   Write a function called detree with the type spec:

detree : cTree -> string list

The function should append only the root node value and all the left nodes on the left subtree. Each value in the Node should be concatenated by themselves given number of times.

You can use the above two functions.

with the following sample runs:

```
- detree (NODE("x",2,(NODE("w",1,EMPTY,EMPTY),(NODE("y",3,EMPTY,EMPTY)));
val it = ["w", "xx"] : string list

- detree (NODE("x", 2,
                    (NODE("w",1,NODE("x",3,EMPTY,EMPTY),NODE("y",2,EMPTY,EMPTY))
                    ,(NODE("y",3,EMPTY,EMPTY)));
val it = ["xxx", "w", "xx"] : string list

- detree EMPTY;
val it = [] : string list
```

4. Finally, write a function which inserts strings into the cTree datatype.

Your function should have this type spec: string -> cTree -> cTree

Note: i) the strings should be stored in-order in the tree.
ii) inserting another copy of a string should just increase the count field on that node.