CS119 – Lab 2A
Due Date: February 13

Purpose: Recursion is a powerful tool for for solving algorithmic problems by having a function repeatedly call itself. We start with a version of recursion called Linear recursion where the function makes a single call to itself within the returned expression.

Knowledge: This lab will help you become familiar with the following content knowledge:

- How to write linear recursive functions

Task: Follow the steps in this lab carefully to complete the assignments. Copy the lab2 directory and write all of your functions in the file `Example2.hs`.

Assignment 1:
Look at the function `downUp`. We want this function to behave as follows:

```
> downUp (word "cake")
[cake ake ke e ke ake cake]

>downUp (word "a")
[a]
```

If you try this however, you will notice that there is an error in the definition of `downUp`. Fix the error.

Criteria for Success: The function uses linear recursion and behaves properly for the examples given above.

Assignment 2:
We want a function `countdups` which takes a sentence and returns the number of words in the sentence that are immediately followed by the same word:

```
> countdups (sent "y a b b a d a b b a d o o")
3

>countdups (sent "yeah yeah yeah")
2
```

Write a linear recursive function `countdups`.

Criteria for Success: The function uses linear recursion and behaves properly for the examples given above.

Assignment 3:
We want a function `explode` which behaves as follows:

```
> explode (word "dynamite")
[d y n a m i t e]
```

The function takes a word and returns a **sentence** containing the single letters of this word. You do not have to worry about adding spaces since a sentence already has spaces between the words.

Criteria for Success: The function uses linear recursion for the task and returns the correct type, which is a **sentence rather than a word**.

Assignment 4:
Define a function `stackCopies :: Int -> Image -> Image` such that `stackCopies n q` will produce a stack of n copies of the quilt image q.

Hint: Your base case will have to be when there is one copy of the image q.

Criteria for Success: The function correctly stacks **any** quilt for an n value greater than 0 using linear recursion.

Assignment 5:
Define a function `quilt :: Int -> Int -> Image -> Image` in which `quilt w h q` will produce a quilt of width w and height h composed of multiple copies of quilt image q.

Hint: Put `stackCopies` of the image together using `sideByside` using linear recursion.

Criteria for Success: For positive values of w and h a quilt is generated with the proper number of rows and columns.

Just for fun, you can design your own quilts.

Submit your `Example2.hs` file in Canvas for grading.