Problem 1: [20 points] In class we wrote a function that replaced certain letters in a string with ‘_’ characters. Define a function called hide that does something similar, but lets the user pass in a function that determines which characters to hide. For example, the first sample input below is asking hide to replace all characters from the string that are equal to ‘e’. Feel free to use recursion, or not, as you see fit.

```haskell
> hide (=='e') "hello eerie elephant"
"h_llo __ri__ _l_phant"
> hide (<'k') "hello eerie elephant"
"__llo_ r___ _l_p__nt"
> hide (`elem` ['a'..'z']) "When To Finish?"
"W___ T_ F_____?"
```
Problem 2a: [15 points] Write a recursive function called pairedWith that takes an item and a list of two-tuples and returns a list containing everything that shares a tuple with the item. In the first sample run below, for example, the answer is [4,6] because both 4 and 6 are found in tuples with 5.

```
> pairedWith 5 [(1,2),(4,5),(5,6),(6,6)]
[4,6]
> pairedWith 2 [(2,2),(2,3),(10,3),(3,2)]
[2,3,3]
> pairedWith 'w' [('f','w'),('z','x'),('w','u'),('n','w')]
"fun"
```

Problem 2b: [15 points] Solve the same problem again, this time without using explicit recursion.
Problem 3a: [15 points] In the graph coloring assignment you made use of Haskell’s Maybe type, which allows us to elegantly handle situations where there might not be anything to return. Below, define an explicitly recursive function called maybeMap that takes a function and a list of Maybe values, and applies the input function to the Just values in the list. The output from maybeMap should be a corresponding list of Maybe values, as shown below.

```
Main> maybeMap (succ) [Just 2, Just 3, Nothing, Just 5]
[Just 3, Just 4, Nothing, Just 6]
Main> maybeMap (\x->x*x) [Nothing]
[Nothing]
Main> maybeMap (succ) []
[]
```

Problem 3b: [10 points] The expression below uses maybeMap to process a list of terms involving division by zero. Would lazy evaluation be able to evaluate the expression without a division by zero exception occurring? Explain why or why not.

```
length (maybeMap (\x->x) (map (\n->Just (n `div` 0)) [1..10]))
```
Problem 4: [25 points]

The questions below apply to this mysterious function.

\[
mystery \ z \ lst = \text{foldr1} \ help \ lst \\
\text{where} \\
\quad \text{help} \ n \ m = \text{if} \ \text{foo} \ n > \text{foo} \ m \ \text{then} \ n \ \text{else} \ m \\
\quad \text{foo} \ s = \text{length} \ (\text{filter} \ (==z) \ s)
\]

4a) In English, what does the helper function \text{foo} do?

4b) If we could call \text{help} directly with "care" and "rat", what would it return if \text{mystery}'s first argument was 't'?

4c) What does \text{mystery} return if passed 'e' and ["hello", "eerie", "elephant"]?

4d) Describe, in \textit{English}, what the \text{mystery} function does in general.