Functions in Haskell

EECS 368 Homework

Due: Monday 30th April, 10am (start of class)

Qixiang Liu

Create the instructed functions. INCLUDE THEIR TYPES. Create helper functions as needed.

1. Write a Haskell function `zip4` which takes 4 arbitrary lists and zips them together.

```
zip4 :: [a]->[b]->[c]->[d]->[(a,b,c,d)]
zip4 [] _ _ _ = []
zip4 _ [] _ _ = []
zip4 _ _ [] _ = []
zip4 _ _ _ [] = []
zip4 (x:xs) (y:ys) (z:zs) (u : us) = (x,y,z,u) : zip4 xs ys zs us
```

2. Write your own non-recursive `length'` function that uses `map`.

```
length' :: [a] -> Int
length' = sum.map (\_ -> 1)
```

3. Write a function: `nths` which takes a `list` and an `Int` and returns every nth element. Examples:
   a. `nths ['a','b','c','d','e','f'] 1`
      `['a','b','c','d','e','f']`
   b. `nths ['a','b','c','d','e','f'] 2`
      `['b','d','f']`
   c. `nths ['a','b','c','d','e','f'] 3`
      `['c','f']`

```
nths :: Eq a => [a]->Int->[a]
nths xs n = if rest ==[]
            then []
            else head rest : nths (tail rest) n
           where rest = drop (n-1) xs
```

4. Write a function `allPairs` that takes two lists and pairs all elements of the first with all elements of the second. Examples:
   a. `allPairs [1] ['a']`
      `[(1,'a')]`
   b. `allPairs [1] ['a'..'b']`
      `[(1,'a'),(1,'b')]`
   c. `allPairs [1] ['a'..'c']`
      `[(1,'a'),(1,'b'),(1,'c')]`
   d. `allPairs [1..2] ['a'..'c']`
      `[(1,'a'),(1,'b'),(1,'c'),(2,'a'),(2,'b'),(2,'c')]`

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