CS119 – Module 4: Tail Recursion

Purpose: There is a form of recursion, called tail recursion, that feels more like iteration (loops). There are times when this type of recursion may be more natural for solving a problem.

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

- Tail recursion

Activity: With your group perform the following tasks and answer the questions. You will continue to use the lab2 files. You will be reporting your answers back to the class in 45 minutes.

1. Take a look at the following function and complete the substitutions to determine the value of the expression `f 3`

   ```
   f::Int -> Int
   f n = fTail n 1 where
         fTail m result =
             if m==0
             then result
             else fTail (m-1) (2*m+result)
   ```

   The `where` allows us the define a helper function inside our function and the expression for `f` is just a call to the helper function.

   ```
   f 3
   fTail 3 1
   fTail 2 ____________________
   fTail 1 ____________________
   fTail 0 ____________________
   _______________
   ```

   No winding and unwinding! This should *feel* more like looping with the variables `m` and `result` changing each time. It is still recursion though!

2. Use the substitution model to determine the value of the expression `g (word "abc")`

```
g::Language -> Language
g wd = gTail wd (word "") where
   gtail w result =
      if (empty w)
          then result
          else gTail (butFirst w) ((firstItem w)+++result)

g "abc"
gTail "abc" ""
gTail ___________________   ___________________
gTail ___________________   ___________________
gTail ___________________   ___________________
___________________
```

3. All tail recursive functions have a helper function with an extra parameter in which you build up the result. The result is returned at the base case.
Complete these tail recursive functions

```
power: Int -> Int -> Int
power base exp = pTail base exp _________ where
  pTail b e result =
      if e == 0
         then result
         else pTail b (e-1)  (___________________)

length:: Language -> Int
length wd = lTail  wd _________ where
  lTail w result =
     if (empty w)
           then result
           else lTail  (___________________) (___________________)
```

4. Complete the tail recursive function which sums the digits of an integer n. For example `digitSum 173` would return 11. To assist you I am providing you with the substitution model.

```
digitSum 173
digitTail 173 0
digitTail 17 3
digitTail 1 10
digitTail 0 11
11

lastDigit:: Int -> Int
lastDigit n = n `rem` 10

butLastDigit:: Int -> Int
butLastDigit n = n `div` 10

digitSum:: Int -> Int
digitSum n = digitTail n 0 where
  digitTail n sum =
    if n==0 then sum
    else digitTail ( ___________________) (___________________)
```

Complete the following assignments to be submitted for grading. Each should be done individually but you can consult with a classmate to discuss your strategies or if you get an error message that you do not understand.
Write all of your functions in the file `Example2.hs`.

Assignment 1:
You previously wrote a function `explode` which behaves as follows:

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

Write a tail recursive function `explodeTail`.

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

Assignment 2:
You previously wrote 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 tail recursive function `countdupsTail`.

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

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