ECS 140B
Homework Assignment #2
Due no later than 6:00pm Friday, April 27, 2018

Your solutions should be submitted as a single .hs (Haskell) file via Canvas. Name your file "asgn2.hs". Any text that is not Haskell code should be commented out. Any Haskell code that is not working Haskell code should also be commented out, with an added explanation about why your code doesn't work. Do all work on your own. This is not a two-person project.

All your Haskell function definitions must include explicit type declarations.

Problem 1

Create a Haskell function called **multiply** which takes two arguments, both non-negative integers, and returns the product of those values. Make sure your function computes the product by repeated addition and uses natural or augmenting recursion (i.e., not tail recursion) to do so. Here are some examples:

```
> multiply 0 4
0
> multiply 4 0
0
> multiply 4 3
12
>
```

Problem 2

Create a Haskell function called **multiply_tr** which takes two arguments, both non-negative integers, and returns the product of those values. Make sure your function computes the product by repeated addition and uses tail recursion to do so. Here are some examples:

```
> multiply_tr 0 4
0
> multiply_tr 4 0
0
> multiply_tr 4 3
12
>
```

Problem 3

Create a Haskell function called **power** which takes two arguments, both non-negative integers, and returns the result of raising the value of the first argument to the power given by the second argument. Make sure your function computes the product by repeated multiplication using the **multiply** function you defined in Problem 1. Your solution should use natural or augmenting recursion (i.e., not tail recursion) to compute the result. Here are some examples:

```
> power 0 4
0
> power 4 0
1
> power 4 3
64
> power 3 4
81
>
```

Problem 4

Create a Haskell function called **power_tr** which takes two arguments, both non-negative integers, and returns the result of raising the value of the first argument to the power given by the second argument. Make sure your function computes the product by repeated multiplication using the **multiply_tr** function you defined in Problem 2. Your solution should use tail recursion to compute the result. Here are some examples:

```
> power_tr 0 4
0
> power_tr 4 0
1
> power_tr 4 3
64
> power_tr 3 4
81
>
```

Problem 5

The sum of the first n terms of the harmonic sequence is defined as

```
1 + 1/2 + 1/3 + 1/4 + 1/5 + ... + 1/n
```

We can rewrite this definition recursively, so that the sum of the first n terms of the harmonic sequence can be defined as:

```
harmonic (n) = 1 if n = 1
             = 1/n + harmonic (n - 1) if n > 1
```

Using Haskell and the information provided above, construct the function **harmonic** that takes one argument, an integer n that is greater than 0, and returns the sum of the first n terms of the harmonic sequence as a real number. You can use Haskell's built-in arithmetic operations; you don't have to use functions you've written as solutions to previous problems. Your solution should use natural or augmenting recursion (i.e., not tail recursion) to compute the result.

Problem 6

Using Haskell and the information provided in Problem 5, construct the function **harmonic_tr** that takes one argument, an integer n that is greater than 0, and returns the sum of the first n terms of the harmonic sequence as a real number. You may use Haskell's built-in arithmetic operations; you don't have to use functions you've written as solutions to previous problems. Your solution should use tail recursion to compute the result.