CMSC 15100 Introduction to Computer Science I
Homework 2

Your answers should be saved in a single .rkt file and submitted via Canvas before the start of the next lecture. At the top of the file for this assignment and all future assignments include a comment,

```
;; Homework 2
;;
;; your name
;; your CNET id
```

If your code doesn't compile, it may not be graded. Please double check to make sure your code runs before submitting!

Problems

1. (0 points) Revisit the functions you wrote in Homework 1, transcribe them into DrRacket, and run the tests. If there are errors, try and correct them. If everything looks right but it still doesn't run, make a Piazza post asking about it. There's no need to submit anything for this problem.

2. (4 points) Normally, your grade is represented by a score between 0 and 100, which we represent in Racket as an `Exact-Rational`. At the end of the quarter, though, we need to convert that score to a letter grade A, B, C, D, or F. Letter grades can be represented in Racket by a `Symbol` by `'A, 'B, 'C, 'D` or `'F`. Write a function `score->letter` that converts the numeric score to a letter grade. Use the following grade cutoffs:

at least 90:	A
at least 80:	B
at least 70:	C
at least 60:	D
between 0 and 60:	F

   If the input is negative, your code should raise an error.

3. (5 points) Write a function `how-many-animals` that has two integer inputs, the number of cats and the number of dogs, and gives you a `String` saying how many of each animal there are. Specifically, the type of this function is `Integer Integer -> String`. If there are zero of some animal, you shouldn't say anything about that animal. For example,

```
(how-many-animals 2 3) ==> "there are 2 cats and there are 3 dogs"
(how-many-animals 1 3) ==> "there is 1 cat and there are 3 dogs"
(how-many-animals 0 10) ==> "there are 10 dogs"
(how-many-animals 0 0) ==> "there are no animals"
```

Make sure your sentence is grammatically correct. If either input is negative, your code should raise an error.

Your code will want to use the built-in functions `string-append` and `number->string`.

4. (5 points)

   (a) The function `expt` takes two arguments x and y, and computes the exponential x^y. Your task is to implement this function when y is a nonnegative `Integer`. That is, you should implement a function `my-expt` with type `Real Integer -> Real`. When the second input is negative, your code should raise an error.

   (b) Using your function from part (a), write a function `sum-of-powers` which takes two inputs n and k, and returns the sum of the first n numbers raised to the power k. Mathematically,

   $$\texttt{sum-of-powers}(n, k) = 1^k + 2^k + \cdots + n^k$$

5. (4 points) Another important function, `length`, is a built-in Racket function that takes in a list and tells you how long it is. Write code for a function `my-length` that replicates the length function. The type declaration of the function should be

```
(: my-length : (Listof Number) -> Integer)
```

6. (6 points) Since a young age, you've been an avid collector of Pokémon cards, and over the years you've managed to get quite a few. To keep track of your collection, you made a `(Listof Boolean)` that represents which of the cards you own. For example, this list

```
'(#t #t #f #f #f)
```

   represents that you have the first and second Pokémon, but you're missing Pokémon three, four, and five. Fortunately, your best friend shares your interests, and has their own `(Listof Boolean)` representing their own collection. You want to know how many Pokémon your friend has that you don't. Make a recursive function `theirs-not-mine` that computes this for you. The type of the function should be `(Listof Boolean) (Listof Boolean) -> Integer`.

   If the two input lists are not the same length, your code should raise an error. You should **not** use the `length` function for this problem, nor should you implement a helper function that checks if the lists have the same length. You should raise the error in the base cases.