Assignment 5

At the beginning of your files, include the line:

```
from typing import *
```

This will allow us to use some of the more complicated types in mypy.

1. 30 points

complete the following function:

```
def map(n : List[int], f : Callable[[int],str]) -> List[str]:
    #your code here
```

map should take a list of the form [x1,x2,x3,...,xn], and a function f, and construct a new list of the form [f(x1),f(x2),f(x3),...,f(xn)]

2. 50 points

Here is some code which computes the fibonacci numbers:

```
def fib(n : int) -> int:
  if n < 2:
    return n
  else:
    return fib(n-1) + fib(n-2)
```

(a) 5 points

For how large a value n can you compute fib(n) before the computation takes more than a few seconds on your computer? (answer doesn't have to be exact)

(b) 30 points

Rewrite the code using a memo to save the result of computations

(c) 5 points
Now what happens as you increase n?
Does the time your code takes increase at the same rate as before?

(d) 10 points
For each version of the code, state whether the time taken for the code is polynomial or exponential in terms of n.

Note that you can write your answers to the response questions in comments if you desire!

3. 20 points

For each of these problems, we imagine writing a solution using recurison.
Given the subproblems (the recursive calls) specified, choose one of:

(a) subproblems do not repeat (tree)
(b) subproblems repeat, but recomputing only harms efficiency
(c) cycles exist, so recomputing can result in non-termination

(i) 5 points
Tic-Tac-Toe, where the subproblems are the result of making every possible move.
(ii) 5 points
Finding a path out of a maze, where the subproblems are
the (potential) paths out following moving to every adjacent square.
(iii) 5 points
The factorial function, where the subproblem of n! is (n-1)!
(iv) 5 points
The fibonacci function, where the subproblems of fib(n) are fib(n-1) and fib(n-2).