Math 308: Bridge to Advanced Math
Programming Assignment 1

Instructions.

- Use IDLE to create a text file entitled `pa1.py` containing three functions, one for each of the problems listed below.
- After you create each function you should test that it works correctly. (Examples of successful input/output were provided with this in mind.)
- Upload this file to blackboard. Do do this, log in to blackboard. Click on "Programming Assignments" in the left panel, then click "Programming Assignment 1". Click on the button "Browse My Computer" and select your `pa1.py` file. Then click the "Submit" button at the bottom of the page.

Grading. You work will be graded based on the following criteria:

- If the file generates an error when loaded into Python, your grade will be a zero.
- I will call the functions listed in the problems with the *exact* names listed below with test cases (similar to the Example Input/Output listed below). Your program should return a correct answer. You will get full credit for that case if it returns a correct answer, and zero credit if it returns a wrong answer.
- The correct response must be returned from each function. (Printing it out will not count.) Use the `return` statement.

A file with the correct format was posted at

`http://wphooper.com/teaching/2015-fall-308/docs/pa1/pa1.py`

(You can just edit the contents of the functions.)

Problems. Inside the file `pa1.py` place three functions as described below which solve each of the following problems. Only include these three functions, and be sure to title them as described in the problems. (Improperly titled functions will not be called properly.) To write these functions, it should be sufficient to understand the Basic mathematics document on the course programming page.

1. Write a function `quadratic_maximum(b,c)` which takes two numbers b and c. It should return the maximal value of the function $f(x) = c + bx - x^2$.

   Examples of successful input/output:

```
>>> quadratic_maximum(0,10)
10.0
>>> quadratic_maximum(2,0)
1.0
>>> quadratic_maximum(5,13)
19.25
```

2. An integer $p \geq 2$ is *prime* if the only positive numbers which divide it evenly are itself and one.

   Write a function `is_prime(p)` which takes as input an integer $p \geq 2$ and returns `True` if p is prime and `False` if p is not prime.

   Hints: The number p is not prime if and only if there is an integer n with $2 \leq n < p$ so that the remainder when dividing p by n is zero.

   Example of successful input/output:

```
>>> is_prime(7)
True
>>> is_prime(25)
False
```

3. *Newton's method* is a very efficient way to find a root of a differentiable function f starting with a point near the root. The method gives a sequence $x_0, x_1, x_2, \ldots$ of numbers which rapidly approach the root if the initial point is sufficiently close to the root. The value x_0 is the starting point. Given x_k the value of x_{k+1} by intersecting the x-axis with tangent line to the graph of f at the point $\big(x_k, f(x_k)\big)$. That is,

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

   An illustration of this process is shown at the end of this question.

   Write a function `newtons_method(f,df,x0,n)` which takes as input a function $f : \mathbb{R} \to \mathbb{R}$, its derivative $df = f'$ (also a function from $\mathbb{R}$ to $\mathbb{R}$), an initial point x_0 and an integer $n \geq 1$. The function should return the value x_n obtained by iterating Newton's method n times.

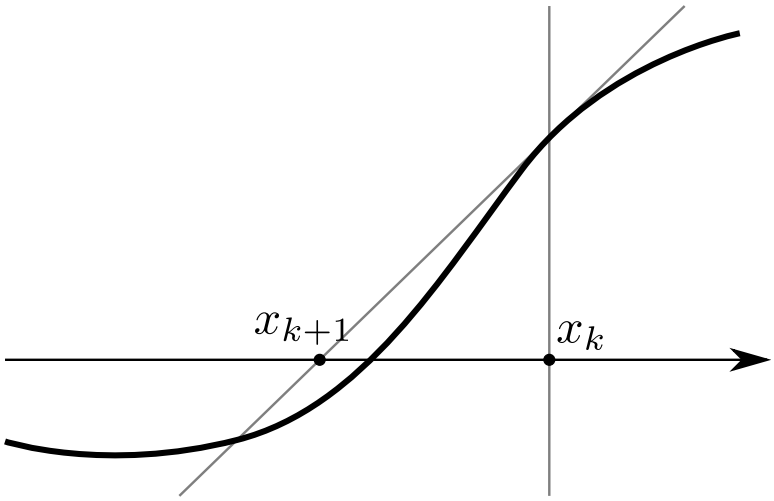
   Example of successful input/output:

```
>>> def f(x):
return 2-x**2

>>> def df(x):
    return -2*x

>>> newtons_method(f, df, 1, 10)
1.414213562373095

>>> newtons_method(math.sin, math.cos, 3, 10)
3.141592653589793
```


x_{k+1}
x_k