Last name _____

First name _____

LARSON—MATH 255–CLASSROOM WORKSHEET 09
Intermediate Value Theorem.

1. Log in to your Sage/Cocalc account.

   (a) Start the Chrome browser.

   (b) Go to `http://cocalc.com` and sign in.

   (c) You should see an existing Project for our class. Click on that.

   (d) Click "New", call it **c09**, then click "Sage Worksheet".

   (e) For each problem number, label it in the Sage cell where the work is. So for Problem 1, the first line of the cell should be `#Problem 1`.

   If $f(x)$ is a continuous function, and $f(a) \leq c \leq f(b)$ then there is some real number x in the interval $[a, b]$ where $f(x) = c$ (that's the **Intermediate Value Theorem**). We will define a function that finds this x. We will do this in steps.

2. Given a continuous function $f(x)$, and numbers a, b and c, define a function `check_conditions(f,a,b,c)` that returns True if $f(a) \leq c \leq f(b)$ and False otherwise. Evaluate.

3. Let $f(x) = x^2$. Evaluate `check_conditions(f,1,2,3)` and `check_conditions(f,1,2,5)`. Is the output what you expected?

4. Given a continuous function $f(x)$, and numbers a, b and c, define a function `test_average(f,a,b,c)` that returns the tuple $(a, (a+b)/2)$ if $f((a+b)/2) \geq c$ and returns $((a+b)/2, b)$ if $f((a+b)/2) < c$.

5. Type and evaluate the following code. It first tests if a given function meets the conditions of the *Intermediate Value Theorem* (IVT). If so this means the desired x is in the interval $[a, b]$. If it does then it successively splits this interval to find out which half the x lives in. The version below does 10 splits (so the produced solution must be within $2^{-10}(b-a)$ of the correct x).

```
def IVT(f,a,b,c):
    if check_conditions(f,a,b,c)==False:
        print "The conditions of the IVT are not satisfied"
    else:
        for i in [1..10]:
            (a,b)=test_average(f,a,b,c)
    return a
```

   $f(x)$ should still be the squaring function. Evaluate `f` to check. If not, let `f(x)=x**2`.

6. Find `IVT(f,1,2,2)`. This will give you an approximation of the square root of 2. Square your result to check how good the produced answer is.

7. Find `IVT(f,1,2,3)`. This will give you an approximation of the square root of 3. Square your result to check how good the produced answer is.

8. Modify the last program to do 20 iterations. Evaluate.

9. Find `IVT(f,1,2,2)` again. Square your result to check how good the produced answer is.

10. Find `IVT(f,1,2,3)` again. Square your result to check how good the produced answer is.

Now we will find this Intermediate Value using *Newton's Method*.

11. Evaluate the following code.

```
def IVTnewton(f,a,b,c):
    fprime=f.derivative()
    if check_conditions(f,a,b,c)==False:
        print "The conditions of the IVT are not satisfied"
    else:
        for i in [1..3]:
            a=a+(c-f(a))/fprime(a)
    return a
```

12. Find `IVTnewton(f,1,2,2)`. This will give you a different approximation of the square root of 2. Square your result to check how good the produced answer is.

    Find `IVTnewton(f,1,2,3)`. This will give you a different approximation of the square root of 3. Square your result to check how good the produced answer is.

13. Modify the last program to do 5 iterations. Retry all your previous uses of this function and compare the outputs.

14. Find `IVTnewton(f,1,2,2)` again. Square your result to check how good the produced answer is.

15. Find `IVTnewton(f,1,2,3)` again. Square your result to check how good the produced answer is.

Step Functions and Scatter Plots

Given a list L of pairs (x, y) you can plot the *step function* that holds y constant from one x to the next with `plot_step_function(L)`.

16. Try `plot_step_function([(x,x) for x in [3..9]])`

17. Try `plot_step_function([(i,sin(i)) for i in [5..20]])`

18. Try `plot_step_function([(i*.2,sin(i*.2)) for i in [5..100]])`