Computer Exercise 3.1.3

This program will attempt to solve $tan(x) = x$ on the intervals $I_1 = [4, 5]$ and $I_2 = [1, 2]$ using the bisection algorithm. For each attempt, there will be a set maximum amount of iterations to prevent the program from blowing up my computer just in case.

Define $f(x) = tan(x) - x$; here, the endpoints of both I_1 and I_2 yield positive and negative pairs as is required to initiate the bisection algorithm.

```
f=@(x) tan(x)-x; %define function to use for bisection method
```

Now, knowing the properties of $tan(x)$, we know that it is undefined for $x = \pm\frac{(2n-1)\pi}{2}$ $\forall n \in \mathbb{N}$. Particularly, we have $\frac{3\pi}{2} \in I_1$ and $\frac{\pi}{2} \in I_2$, so $f(x)$ is not continuous on these intervals, thus giving us no guarantee of a root by the intermediate value theorem. Therefore, the bisection algorithm may or may not yield a root.

```
% attempt on interval I_1

a=4;
b=5;
n=100;
error = 10^(-5);

root1 = bisect(f, a, b, n, error);

fprintf(['Calculated root from bisection for the interval [4,5] is r = %f which ' ...
    'evaluates to f(r) = %f;\nthis yields an error of %f'], root1, f(root1), abs(f(root1)) )
```

```
Calculated root from bisection for the interval [4,5] is r = 4.493409 which evaluates to f(r) = -0.000006;
this yields an error of 0.000006
```

It turns out that applying the bisection method for $f(x)$ on I_1 yields a root.

```
% attempt on interval I_2

a=1;
b=2;
n=1000000;
error = 10^(-2);


root2 = bisect(f, a, b, n, error);
```

```
bisection algorithm was unsucessful after 1000000 iterations; error = 0.557408
```

```
fprintf(['Calculated root "r" from bisection for the interval [1,2] is r = %f' ...
```

```
    ' which evaluates to f(r) = %f'], root2 , f(root2))
```

```
Calculated root "r" from bisection for the interval [1,2] is r = 1.000000 which evaluates to f(r) = 0.557408
```

Wow. After a whopping one million iterations, the bisection algorithm never managed to yield an error less than 0.01. Indeed, the results of the bisection algorithm for discontinuous functions can either yield a root or diverge. If I did not set a maximum amount of iterations, my computer might've caught on fire or something.

```
function c=bisect(f, a, b, n, error)
c=(a+b)/2;
i=1; %set iteration counter
%error = |f(c) - 0|
    while (abs(f(c))>error) && (i<=n) %exit while loop once error tolerance
       %  or max iterations has been reached
        if f(a)<0 && f(c)<0 %if true, then f(a)f(c)>0
            a=c;
        else % otherwise, f(b)f(c)>0
            b=c;
        end
        c=(a+b)/2;
        i = i+1; %update iteration counter
    end
    if (abs(f(c))>error) %if this triggers, this means that max iterations
       %  has been reached before error tolerance
        fprintf(['bisection algorithm was unsucessful after %d iterations; ' ...
            'error = %f'], i-1, abs(f(c)))
    end
end
```