Due: Thursday January 28th, 2010
Problems to be turned in 2, 3, 4

1. Evaluate the following by hand and check your results in OCTAVE

   (a) 5|4

   (b) ˜3

   (c) Given x = [0 5 3 7] and y = [0 2 8 7], u = x((x>y) & (x>4)).

   (d) A = `reshape`(1:8,2,4); B = A (5*ones(2,2))

2. Write the OCTAVE statements that use a loop and the `fprintf` function to produce the following table(the format of the numerical values should agree exactly with those printed in the table):

`theta`	`sin(theta)`	`cos(theta)`
0	0.0000	1.0000
60	0.8660	0.5000
120	0.8660	-0.5000
180	-0.0000	-1.0000

3. `myage`

   (a) Write down a function `myage`, that returns your age in years.

   (b) Extend the above `myage` function with an optional input argument, `onDate`, can be provided for testing. This modified `myage` function should return your age in years on `onDate`, where `onDate` is a OCTAVE date number. (see help datenum). Your program should correctly compute your age on the following test dates:

      i. When you first encountered Siva: (Interview date – July 2007).

      ii. One day before your first birthday.

      iii. On the day the assignment is due: January, 31st, 2008.

      Hint: you may use built-in datevec function

   (c) Extend the `myage` function developed so far in the preeceding problem to provide an optional return of the months and days since you were born. For example: Suppose you were born on April, 1, 1904. The function should yield the following values:

```
> > y = myage(datenum(2004,7,11))
y =   100
>> [y,m] = myage(datenum(2004,7,11))
y =   100
m =   3
>> [y,m,d] = myage(datenum(2004,7,11))
y =   100
m =   3
d =   10
```

4. Spot the bug in the function `bugbreak(n)` below and fix it without dismantling the `break` command:

```
function bugbreak(n)
x = rand(1,n)
k =1 ; while k <= n
    if x(k) > 0.7
        break
     end
     k = k + 1;
end
fprintf('x(k) = % f for k = % d n = % d \ n', x(k), k, n);
```

5. `newsqrt`

   (a) Write a `newsqrt` function to compute the square root of a positive number, using Newton's method, using `while.. end`. An upper limit on the number of iterations and an acceptable error value should be include in the input arguments. Allow for default values if the user decided not to give any or one of the input parameters. (`Do not use any input prompt command.`)

   (b) Write a `newsqrt` function to compute the square root of a positive number, using Newton's method but this time using `for.. end`.

Summary of Week three

At the end of this week you should be able to

1. Write meaningful comments to Octave code.
2. Use indentation and whitespace to make code easier for humans to comprehend.
3. Write a prologue that responds to a request for on-line help.
4. Subdivide a programming project into modules.

After mastering the bare essentials you should move on to a deeper understanding of the fundamentals. Doing so involves being able to

1. Create solutions to substantial programming projects with top-down-design.
2. Use defensive programming strategies to check for obvious errors in input parameters.
3. Use the built-in `error` and `warning` functions to provide feedback for run-time errors.
4. Able to trace logic and numerical errors in Octave code.