CS005 Lab 4:

Use good variable names, and comment your code carefully. We will deduct points otherwise. *Read* the most recent slides, have them ready to show the TA when you ask him questions.

Part 1:

Write a function `IsEven` that takes in an integer, and returns true ('1') if it is even, otherwise returns a false ('0'). This function is nearly *identical* to the `IsOdd` function we wrote in lecture. Read the lecture slides again!

Test your function, by using some code like the code below.

```
EDU>> IsEven(12)
ans =
     1
```

```
EDU>> BobsAge = 15
EDU>> IsEven(BobsAge)
ans =
     0
```

Part 2:

Write a function called `RobustIsEven`. It will be identical to the above function, except if it is called with a number that has a fractional part, it will give an error message and return an `NaN`. Test it carefully as below. **HINT**: Do one of the below

1) Add the code that tests for the fractional part, *after* you test for evenness
2) Use nested if/else statements

```
EDU>> IsEven(12)
ans =
     1
EDU>> IsEven(12.87)
NOT defined!
ans =
     NaN
```

```
EDU>> BobsAge = 15.4
EDU>> IsEven(BobsAge)
NOT defined!
ans =
     NaN
EDU>> IsEven(round(BobsAge))
ans =
     0
```

Part 3:

Write a function called `FiveTimesTable()`. This function is nearly identical is `SevenTimesTable()` in the lecture notes. It will display the five times table for numbers from zero to ten (not 1 to ten).

Part 4:

Write a function called `FiveTimesTable_KtoL()` (Before you begin, read the `CountFromKuptoL(K,L)` example in the slides again). This function prints a subset of the five times table.

```
EDU>> FiveTimesTable_KtoL(3,6)
Five times
   3
 is
   15
Five times
   4
 is
   20
Five times
   5
 is
   25
Five times
   6
 is
   30
```

```
EDU>> FiveTimesTable_KtoL(0,2)
Five times
   0
 is
   0
Five times
   1
 is
   5
Five times
   2
 is
   10
```

Part 5:

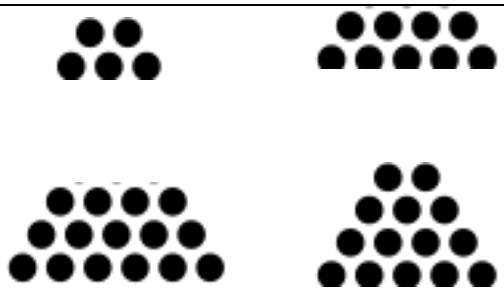
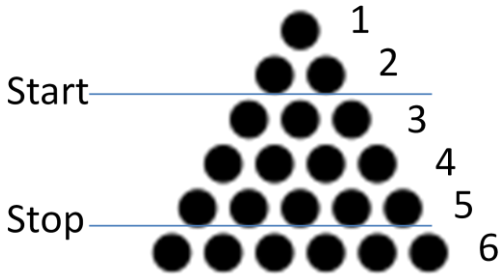
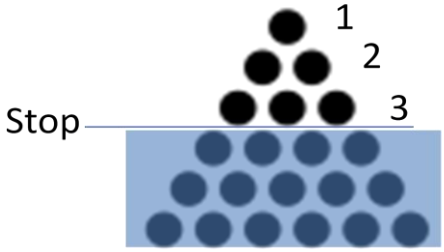
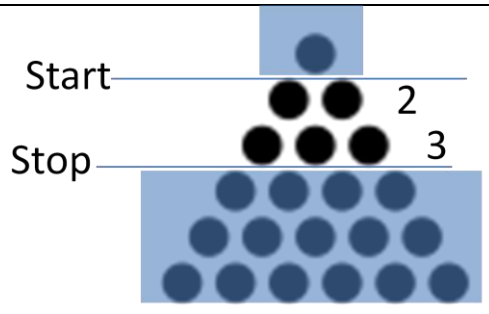
Write a function `GetTruncatedTriangularNumber` that gets the truncated triangular number (see below).

HINT: Here is the first line

```
function TrucTriNum = GetTruncatedTriangularNumber(Start,Stop)
```

Here is how we will use it

```
EDU>> GetTruncatedTriangularNumber(1,3)
ans =
5
```

Look at the four "stacks" of dots to the right.	
Note that they are all special subsets of triangular numbers.	
For example, to build the top left one (the one with 5 dots), we could build a triangular number with `Stop` equal to 3....	
Then we could subtract the triangular number with `Start` equal to 1....	

Hint: If you code up the `GetTriangularNumber` function we wrote in lecture, then you can do this problem in one line of new code. However, if you take more code, that is fine.