Topic: Intro to programming with X86_64 assembler
Points: 25
Due: TBA, using the **turnin** script. ***Submit late at your own risk.***
Overview: In this project, you will write a simple assembly language program

Sometimes it seems that mathematicians have waaayyy too much time on their hands. Incredible volumes of properties have been identified for numbers, some simple to understand, like a prime number, and some a bit more difficult. Some seem important, and others leave the impression that someone needs a life.

For your first project, you will write a program in X86_64 assembler to build an integer with consecutive digits. A C++ version of this program is in the box, and also in the public area for this project on acad as well as on the web and maybe in D2L.

Your program will prompt the user for an initial digit and a number of digits. It then builds an integer meeting those specifications. Some examples are shown in the table.

Initial Digit	#Digits	Number
4	5	45678
7	3	789
8	6	890123

You will need to write a loop to iterate #digits times, and conditionals to error check that the value of start is one digit.

```cpp
// File: cd.cpp
// Build consecutive digit integer

#include <iostream>

using namespace std;

int main()
{int start,digits;
 cout << "Give the first digit >";
 cin >> start;
 // Error check
 if (start < 0 || start > 9) {
        cout << "Error:  Bad Starting digit\n"
        return(-1);
 }
 cout << "How many digits in the number? >";
 cin >> digits;

 // Build the number
 int number=start++;
 for (int i=1;i<digits;i++) {
        // append the new digit
        number=(number*10)+start;
        start=(start+1)%10;
 }
 cout << "The number, with " << digits <<
      " digits, starting from " << start <<
                        " is " << number << endl;
}
```

Your assembler program will not follow the C++ code, as you will use branches for the conditionals and loop. In particular, the conditional can't be done with one branch. Check for too big, and if not, check for too small. If not, continue, but if either too big or too small, branch to the end.

Notes:
- The modulus, or remainder operation is handled by calling *divq* or *idivq* and accessing the remainder in %rdx.
- You must use id blocks and comments. Indent consistently. Preferred indentation is shown in the box. Note the lining up of the instruction (which is also indented), and then the first operand. Only labels may appear against the left margin. Write a program that you would be proud to see posted in the hallway.

Turnin:

Your assembler program, named **p2.m** . You may turn

```
Sum:    mov    10,%l4         !put const 10 into register for comparison
        cmp    %l5,%l4        !compare register values
        bne    loop           !if reg values were unequal (ctr!=10), iterate
        nop                   !delay slot
```

in a file named *readme.txt* (all lower case) to describe design decisions, known bugs, limitations, etc;