HOMEWORK #2 – MATH 5405
SPRING 2016

DUE: THURSDAY, 2/4/2016

1. Find all the irreducible degree 2 polynomials in $(\mathbb{Z}/3)[x]$.
2. Find the multiplicative order of x in $(\mathbb{Z}/2)[x]/(x^4+x+1)$.
3. Consider the polynomials $g(x) = x^3 + 3x + 2$ and $p(x) = x^4 + x - 1$ in $(\mathbb{Z}/5)[x]$.
   (a) Show that the polynomial $g(x)$ is irreducible.
   (b*) Show that $p(x)$ is irreducible too. Try to be smart to avoid a huge amount of brute force work.
   (c) Find the inverse of $g(x)$ in $(\mathbb{Z}/5)[x]/p(x)$ by using the Euclidean Algorithm.

We discussed columnar transposition in class. Since this is not in the book, let me remind you how this works. Say we want to encrypt the message `the german forces are here` using the encryption key `CAT`

C^2	A^1	T^3
t	h	e
g	e	r
m	a	n
f	o	r
c	e	s
a	r	e
h	e	r
e	y	b

We then read the columns vertically starting from the column with first letter in alphabetical order. In this case we get `H E A O E R E Y T G M F C A H E E R N R S E R B`.

4. I created the cipher text `ATTVNYEFWNOHEDTQOOEREIARGSTTWIUSVIATSLEREHURHHEFNEENLY` using columnar transposition and the cipher key `NUMBER`. Decrypt it by hand!
5. The following text was encrypted using a columnar transposition cipher with key length 4.

   `ECOOMVHZGQKWXPEEYPTUBNJSRLDHIRFUOTAO`

   Break the encryption by hand!

There is one more problem on the next page.

6. Using the python code below, I wrote a program to decide if a the group units modulo n has a primitive root/generator. Look at the data it produced below. Figure out the pattern *without googling this on the internet!*

   Hint: The answer is determined by how n factors.

	[2, False]	[3, True]	[4, True]	[5, True]	[6, True]	[7, True]	[8, False]	[9, True]	[10, True]
[11, True]	[12, False]	[13, True]	[14, True]	[15, False]	[16, False]	[17, True]	[18, True]	[19, True]	[20, False]
[21, False]	[22, True]	[23, True]	[24, False]	[25, True]	[26, True]	[27, True]	[28, False]	[29, True]	[30, False]
[31, True]	[32, False]	[33, False]	[34, True]	[35, False]	[36, False]	[37, True]	[38, True]	[39, False]	[40, False]
[41, True]	[42, False]	[43, True]	[44, False]	[45, False]	[46, True]	[47, True]	[48, False]	[49, True]	[50, True]
[51, False]	[52, False]	[53, True]	[54, True]	[55, False]	[56, False]	[57, False]	[58, True]	[59, True]	[60, False]
[61, True]	[62, True]	[63, False]	[64, False]	[65, False]	[66, False]	[67, True]	[68, False]	[69, False]	[70, False]
[71, True]	[72, False]	[73, True]	[74, True]	[75, False]	[76, False]	[77, False]	[78, False]	[79, True]	[80, False]
[81, True]	[82, True]	[83, True]	[84, False]	[85, False]	[86, True]	[87, False]	[88, False]	[89, True]	[90, False]
[91, False]	[92, False]	[93, False]	[94, True]	[95, False]	[96, False]	[97, True]	[98, True]	[99, False]	[100, False]
[101, True]	[102, False]	[103, True]	[104, False]	[105, False]	[106, True]	[107, True]	[108, False]	[109, True]	[110, False]
[111, False]	[112, False]	[113, True]	[114, False]	[115, False]	[116, False]	[117, False]	[118, True]	[119, False]	[120, False]
[121, True]	[122, True]	[123, False]	[124, False]	[125, True]	[126, False]	[127, True]	[128, False]	[129, False]	[130, False]
[131, True]	[132, False]	[133, False]	[134, True]	[135, False]	[136, False]	[137, True]	[138, False]	[139, True]	[140, False]
[141, False]	[142, True]	[143, False]	[144, False]	[145, False]	[146, True]	[147, False]	[148, False]	[149, True]	[150, False]
[151, True]	[152, False]	[153, False]	[154, False]	[155, False]	[156, False]	[157, True]	[158, True]	[159, False]	[160, False]
[161, False]	[162, True]	[163, True]	[164, False]	[165, False]	[166, True]	[167, True]	[168, False]	[169, True]	[170, False]
[171, False]	[172, False]	[173, True]	[174, False]	[175, False]	[176, False]	[177, False]	[178, True]	[179, True]	[180, False]
[181, True]	[182, False]	[183, False]	[184, False]	[185, False]	[186, False]	[187, False]	[188, False]	[189, False]	[190, False]
[191, True]	[192, False]	[193, True]	[194, True]	[195, False]	[196, False]	[197, True]	[198, False]	[199, True]	[200, False]

The following code is also available on the course webpage.

```
import fractions

def eulerPhi(n):          #determines phi(n) (definitely not the best way to do it, it would be better to factor...)
     count = 0
     for k in range(1,n):
          if (fractions.gcd(k,n) == 1):
               count = count + 1
     return count

def isPrimitiveRoot(a,n,phi):       #determines if a is a primitive root mod n, provide the euler phi value as that is the size
                    #note we are not actually doing a good job of this, this is ridiculous brute force
     for i in range(1,phi):
          if ((a**i)%n == 1):
               return False
     return True

def hasPrimitiveRoot(n):       #determines if the group of units mod n has a generator / primitive root
     myPhi = eulerPhi(n)
     for i in range(2, n):
          if (fractions.gcd(i,n) == 1):
               if (isPrimitiveRoot(i,n,myPhi) == True):
                    return True
     return False

def listNumbersWithPrimitiveRoots(limit):      #lists the n from 2 to limit that have a primitive root
     listOfRoots = []
     for j in range(2,limit+1):
          val = hasPrimitiveRoot(j)
          listOfRoots.append([j,val])
          print [j, val]
     return listOfRoots
```