**Mark Eatough**

**CSIS 2430 9:00 Class**

**Programming Project 4**

**RSA Encryption Program**

**Assignment objective:**

Your assignment is to encrypt the message ***"The Queen Can't Roll When Sand is in the Jar"*** using the values for p = 61, q = 53, and e = 17. The associated values of a – z are 0 - 25 respectively. The associated value of (“) is 26, (space) is 27 and (‘) is 26.

**What Worked?:**

Algorithms one and 5 from the previous assignment worked great to help with the encrypting algorithm. To assign values to my characters I created a list of all of the characters that might be used and then used the list indexes associated with those characters to call their values. This seemed to work fairly well.

**What did not work?:**

When grouping my character, I wanted all of their associated values to be two characters long, ie a = 00, b = 01, etc. This seemed to work fine, but when converting these string snippets to integers, I could not have a leading zero, so I had to wait until all of the encryption conversions and calculations were done to format with the necessary leading zeros.

**Comments:**

For this assignment I already knew how two of the important algorithms worked and how to implement them. So all I had to do was figure out how to apply them to accomplish RSA encryption and how to assign the necessary values to my letters and other necessary characters.

'''  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
 \* Discrete Structures  
 \* RSA Encryption Program  
 \* Programmer: Mark Eatough  
 \* Course: CSIS 2430   
 \* Created September 22, 2013  
  
\*This program encrypts a progammed in phrase using   
 \*programatically given encryption values. The output is  
 \*in string form, with spaces separating between every fourth  
 \*number  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
'''  
#may not use any of this for this assignment, but will probably be needed for ceasar cypher  
  
#Create list of characters  
characters = []  
#populate list of characters  
a = ord('a')  
z = ord('z')  
for letter in range(a, z+1):  
 characters.append(chr(letter))  
characters.append('"')  
characters.append(" ")  
characters.append("'")  
  
#first prime number  
p = 61  
#second prime number  
q = 53  
#prime number that is not a divisor of (p-1)\*(q-1)  
e = 17  
#n is used as modulus for public and private keys  
n = p\*q  
#phrase to be encrypted  
phrase = "\"The Queen Can't Roll When Sand is in the Jar\""   
  
#algorithm 1 out of book, used to find binary representation of my exponents  
def makeBinary(n):  
 b = 2  
 q = n  
 binary = ""  
 while(q > 0):  
 a = q%b  
 q = q/b  
 binary = str(a) + binary  
 return binary  
  
#algorithm 5 out of book, used to find modulus of very large numbers  
def findModulus(number):  
 x = 1  
 exponent = 17  
 modulus = n  
 binary = makeBinary(exponent)  
 i = len(binary)-1  
 while (i > 0):  
 number\*=number  
 nmumber = number % modulus  
 if(binary[i-1] == '1'):  
 x\*=number  
 x = x % modulus  
 i-=1  
 return x  
  
#method to convert the characters into their associated values  
#ie a = 0, b = 1 etc.  
def charToIntVal(c):  
 for i in range(len(characters)):  
 if(c.lower() == characters[i]):  
 myChar = str(i)  
 if(i%100 < 10):  
 myChar = "0" + myChar  
 return myChar  
  
#method to convert the associated value to the encrypted value   
def encrypt(p):  
 tempInt = int(p);  
 encryptedInt = findModulus(tempInt)  
 return encryptedInt  
  
#method that parses out the string, converts the characters to numbers,  
#converts those values to encrypted values, and then returns them in  
#string form  
def parseString(s):  
 tempString = ""  
 tempChar = ""  
 newString = ""  
 for i in range(len(s)):  
 tempString += charToIntVal(s[i])  
 for j in range(len(tempString)):  
 tempChar += tempString[j]  
 if(j%4 == 3):  
 if(len(str(encrypt(tempChar))) < 4):  
 k = 0  
 while(k < 4 - len(str(encrypt(tempChar)))):  
 newString += "0"  
 k+=1  
 newString += str(encrypt(tempChar))  
 newString += " "   
 tempChar = ""   
 print newString  
   
  
print phrase  
  
parseString(phrase)

