Izaan
P20-0613

```
from random import randint, getrandbits
from utils.isprime import isPrime
from utils.isCongruentNumber import isCongruentNumber
from utils.coprime import coprime

class BBS:
    p = 0
    q = 0
    n = 0
    seed = 0
    generated values = []

    def __init__(self, p, q):
        self.setP(p)
        self.setQ(q)
        if (self.p > 0 and self.q > 0):
            self.__setN()
            self.__setSeed()

    def setP(self, p):
        if (not self.__checkParams(p)):
            self.p = p

    def setQ(self, q):
        if (not self.__checkParams(q)):
            self.q = q

    def __checkParams(self, number):
        isError = False
        if (not isPrime(number)):
            print(number, 'is not Prime')
            isError = True
```

```
def _set N (self):
    self.n = self.p * self.q

def _setSeed (self):
    while (not coprime (self.n, self.seed) and self.seed)

def _generateValue (self):
    if (self.p > 0 and self.q > 0)
    x = 0
    while (not coprime (self.n, x)):
    x = randint (0, self.n)
    return pow (x, 2) % self.n

def generateBits (self, amount):
  if (self.p == self.q)
    print ('p shall be different than q')

  if (self.n == 0):
    print ('N is equal 0')

  else:
      bitsArray = []
      amount += 1

  for i in range (amount):
      generatedValue = self._generateValue()
      self.generatedValue.append (generatedValue)
        if (generatedValue % 2) == 0):
          bitsArray.appal (0)
        else
            bitsArray.appen (1)
```