JUNE 21ST MATH PROBLEM SET

I have had my results for a long time: but I do not yet know how I am to arrive at them. – Carl Friedrich Gauss.

In Sage, I wrote the following code.

```
def isGeneratorMod(x,n): #i wrote this in the most naive way possible,
                         #it assumes x and n are relatively prime
                         #otherwise it will return the wrong answer
    y = 1
    b = euler_phi(n)
    for i in range(0, b-1):
        y = (y*x)%n #just run through powers of x and see if I get 1
        if (y == 1): #I got 1 too soon, not a generator
            return False
    return True #I never got 1, it is a generator

def isCyclic(n):  #returns true if there is a generator mod n,
                  #otherwise it returns false
    for i in range(1,n):
        if (gcd(i,n) == 1):  #check if it might possibly be a generator first
            bb = isGeneratorMod(i,n) #finds if i is a generator mod n
            if (bb == True):
                return True
    return False

def findCyclicN(maxN):
    for i in range(2, maxN):
        bb = isCyclic(i)
        print str(i) + ": " + str(bb)
```

Then I ran `findCyclicN(200)`. The output of which is on the next page.

1. Your job is to figure out for which n do the set of numbers relatively prime to n have a generator (are *cyclic* is the technical term). 2 points to the two fastest teams to spot the pattern.

Hint: Write down the prime factorizations of the integers n.

2: True
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

Don't start on this until all the teams found the pattern. See if you can figure out why the pattern holds (this is hard, and relies on the Chinese remainder theorem).

Another question is, how many generators are there? I wrote the following code which should could the number of generators.

```
def countGenerators(n):
    counter = 0
    for i in range(1,n):
        if (gcd(i,n) == 1):
            bb = isGeneratorMod(i,n) #finds if i is a generator mod n
            if (bb == True):
                counter = counter + 1
    return counter

def listGeneratorsCount(maxN):
    for i in range(2, maxN):
        genCount = countGenerators(i)
        if (genCount > 0):
            print  str(i) + ": " + str(genCount)
```

Here is my output when running `listGeneratorsCount(200)`.

```
2:  1          34:  8          82:  16          139:  44
3:  1          37:  12         83:  40          142:  24
4:  1          38:  6          86:  12          146:  24
5:  2          41:  16         89:  40          149:  72
6:  1          43:  12         94:  22          151:  40
7:  2          46:  10         97:  32          157:  48
9:  2          47:  22         98:  12          158:  24
10:  2         49:  12         101:  40         162:  18
11:  4         50:  8          103:  32         163:  54
13:  4         53:  24         106:  24         166:  40
14:  2         54:  6          107:  52         167:  82
17:  8         58:  12         109:  36         169:  48
18:  2         59:  28         113:  48         173:  84
19:  6         61:  16         118:  28         178:  40
22:  4         62:  8          121:  40         179:  88
23:  10        67:  20         122:  16         181:  48
25:  8         71:  24         125:  40         191:  72
26:  4         73:  24         127:  36         193:  64
27:  6         74:  12         131:  48         194:  32
29:  12        79:  24         134:  20         197:  84
31:  8         81:  18         137:  64         199:  60
```

2. Find a pattern for how many generators there are.

Hint: Compare the number of generators mod n to $\phi(n)$.