Student Name: Jonathan Riordan

Student ID: C13432152

Part 1

Program to generate a sudo random sequence. The algorithm itself works by setting a variable xi to 9 at initiation. x1 is a global variable. we return xi after xi multiplies itself and mod m. “m” hold the number 11 by 19. We access the function 12 times so we get a sequence of numbers that have a random sequence.

Output of code:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Part 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

81

82

36

42

92

104

157

196

169

137

168

9

Code:

import r4nd0m

print "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Part 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

m = 11 \* 19

xi = 9

def rng():

global xi

xi = (xi \* xi) % m

return xi

for i in range(12):

print rng()

Part 2

If p is greater than 0.01, this means the sequence is random. From the output below, the conclusion is that the p value is 0.05, this means the sequence is random. To make the sequence more random, we use bigger numbers for xi and m. In order to run the following code, it has to be run using python 3 and a library RandomnessTests is used.

Output of school:

81

82

36

42

92

104

157

196

169

137

168

9

p value

0.563702861651

Code

'''

Advanced Security

Student Name: Jonathan Riordan

Student ID: C13432152

Lab 5 - Part 2

'''

from RandomnessTests import RandomnessTester

m = 11\*19

xi = 9

def rng():

global xi

xi = (xi\*xi) % m

print(str(xi))

return xi

bitstr = ""

for i in range(12):

bitstr += bin( rng() )[-1:]

rng\_tester= RandomnessTester(None)

p = rng\_tester.monobit(bitstr)

print("p value ")

print(p)