Question 2)

Note that the sample code for Miller Rabin returns “True” if the test finds, conclusively, that *n* is composite, otherwise the function returns “False.” To indicate that the function did not find anything conclusively. As noted in this book, we can decide with high probability if *n* is prime or composite if we run this test several times. This exercise is to implement a version of the Miller Rabin test that does so.

1. Implement a function that performs the “witness procedure” of Miller Rabin. I.E. The code that checks whether or not *a* in {1,2,…,n-1} has the specified properties.
2. Use the function that you wrote for part (a) to implement a function that takes a positive integer n (> 2), and a list of integers in {1,2,…,n-1} and performs the “witness procedure” on each one. If any one of these determines that *a* is composite, then return False (to indicate a is composite) otherwise return True, to indicate that (with high probability) *n* is prime.

Question 2 - Solution

Note that the sample code for Miller Rabin returns “True” if the test finds, conclusively, that *n* is composite, otherwise the function returns “False.” To indicate that the function did not find anything conclusively. As noted in this book, we can decide with high probability if *n* is prime or composite if we run this test several times. This exercise is to implement a version of the Miller Rabin test that does so.

1. Implement a function that performs the “witness procedure” of Miller Rabin. I.E. The code that checks whether or not *a* in {1,2,…,n-1} has the specified properties.

def MillerRabinWitnessTest(n, q, k, a):

# if a^q mod n == 1 then return inconclusive

if (1 == a^q):

return True

# (3) for j = 0 to k-1 do: if a^(2^j \* q) mod n = n-1 return inconclusive

e = q

for j in xrange(k):

if (n-1) == (a^e):

return True

e = 2\*e

1. Use the function that you wrote for part (a) to implement a function that takes a positive integer n (> 2), and a list of integers in {1,2,…,n-1} and performs the “witness procedure” on each one. If any one of these determines that *a* is composite, then return False (to indicate a is composite) otherwise return True, to indicate that (with high probability) *n* is prime.

def MillerRabinWitnessListTest(n, witnesses):

r"""

Given a list of witnesses a this runs the Miller-Rabin witness test on each one.

Returns True if n is prime with high probability,

otherwise returns false.

"""

R = IntegerModRing(n);

q = n-1

k = 0

while (1 == (q % 2)):

k += 1

q = q.quo\_rem(2)[0] # q/2 but with result of type Integer

could\_be\_prime = True;

for j in xrange(len(witnesses)):

a = R(witnesses[j])

could\_be\_prime = could\_be\_prime and \

MillerRabinWitnessTest(n, q, k, a)

return could\_be\_prime