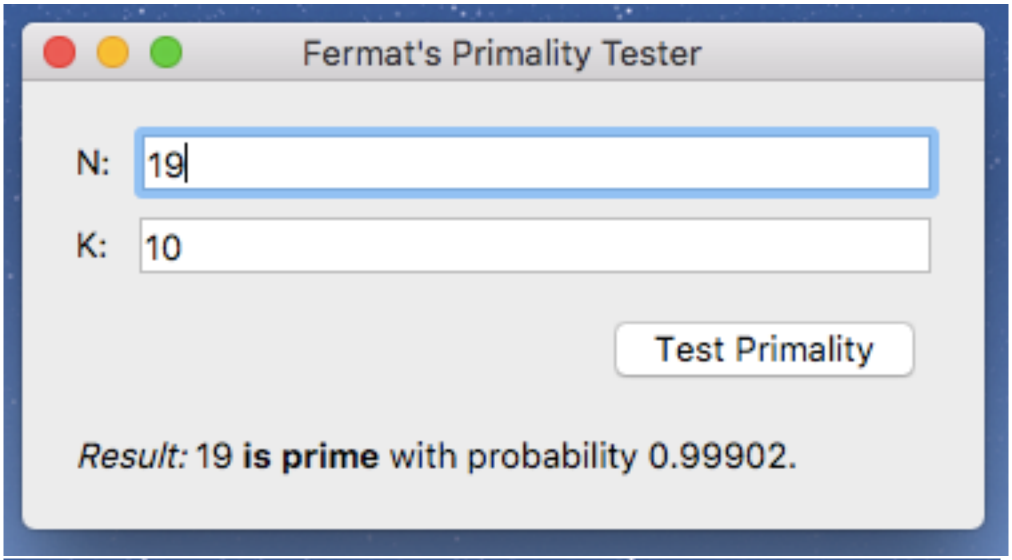
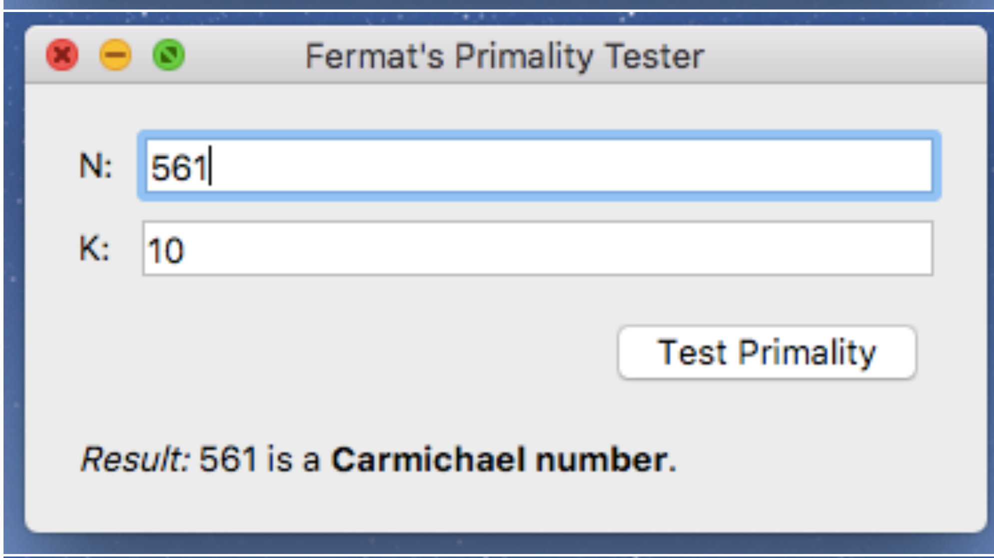
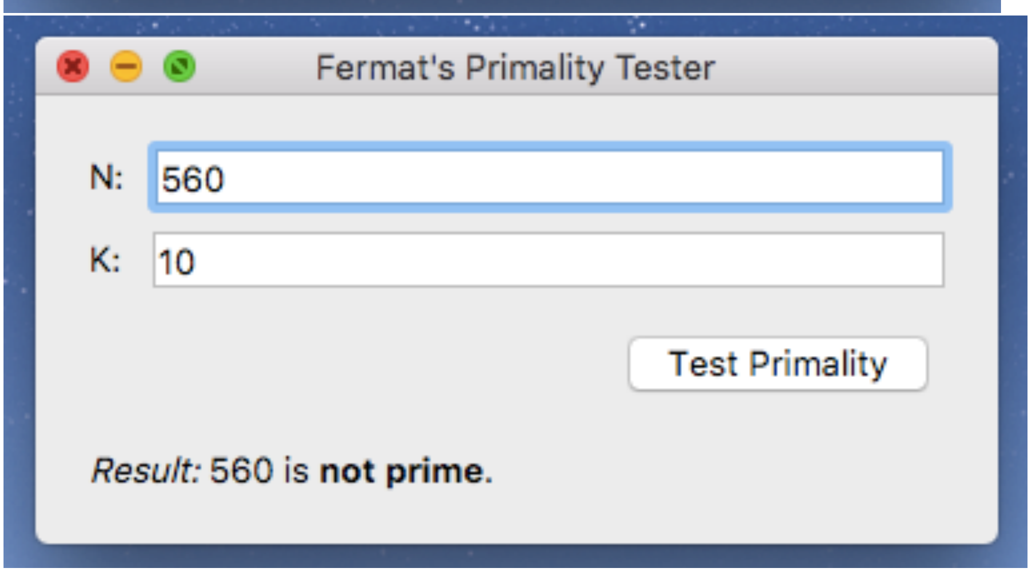
Cody Kesler
CS 312 Section 2 - Ryan Farrell

ScreenShots:







Cody Kesler
CS 312 Section 2 - Ryan Farrell

Code:

```
import random

#Project Total Big O = Big O(n^3) + Big O(n) + Big O(k^2) + Big
O(k*^2n^3) =

********Big O(k^2*n^3)***********

#Big O(k*(n^2 + n^3) = Big O(k^2*n^3)
def prime_test(N, k):
    if(N <= 0): #big O(1)
        return False
    else:
        numbs = list() #big O(1): repeat 1
        for i in range(0, k): #big O(1) repeat k

        #Get a random number that haven't been used before
            rand = random.randint(3, N) #big O(N^2)
            while(rand in numbs):
                rand = random.randint(1, N) #big O(N^2) repeat k
times

        #Tack number onto the list to check for next time
            numbs.append(rand) #big O(1)

#Check to see if it is a prime number
    if(mod_exp(rand, N-1, N) != 1):  #Big O(n)
        #If it is not a prime by the Fermat algorithm then its
composite
        return 'composite'
    else:
        for j in range(0,len(numbs)):
            if not(is_carmichael(N,rand)):  # Big O(n^3) repeat
k times
          #if is not a carmichael(its a prime) then return prime
                return 'prime'
            else:
                #if not it a a carmichael return that
                return 'carmichael'
```

Cody Kesler
CS 312 Section 2 - Ryan Farrell

```
#Big O((n*y^2 + z^2)) = Big O(n)
def mod_exp(x, y, N):
    # You will need to implements this function and change the
return value.
    if(y == 0):  #big O(1)
        return 1  #big O(1)
    z = mod_exp(x, y//2, N)  #big O(1) repeat n times (n = bits
in y)
    if(y % 2 == 0):  #big O(y^2)
        return z**2 % N  #big O(z^2)
    else:
        return x*z**2 % N  #big O(xz^2)

 #Big O(k^2)
def probability(k):
    # You will need to implements this function and change the
return value.
    return (1- (1/(2**k)))  #big O(k^2)

#Big O(y^2 * n + n + n^3) = Big O(n^3)
def is_carmichael(N, a):
    y = N - 1  #big O(1)

    while(y % 2 == 0):
        y = y//2 #big O(y^2): repeat O(n) (n = bits in y)

    result = mod_exp(a, y, N)  #big O(n)
    if(result == 1):  #big O(1)
        return False  #big O(1)

    while(result != 1):  #big O(1)
        prev = result  #big O(1)
        result = result**2 % N  #big O(n^2): repeat O(n)

    if(N - prev == 1): #big O(1)
        return False #big O(1)
    else:
        return True #big O(1)
```

Cody Kesler
CS 312 Section 2 - Ryan Farrell

Time Complexity:
def prime_test(N, k): Big O(k^2*n^3)
def mod_exp(x, y, N): Big O(n)
def probability(k): Big O(k^2)
def is_carmichael(N, a): Big O(n^3)
Thus, the total time complexity: Big O(k^2*n^3)

Space Complexity:
List numbs: (kN)
Recursion of mod_exp() (log(y)*4N)
Probability() (3N)
Is_carmichel() (7N)
Total: (kN+log(y)*4N+3N+7N)

Probability Equation:
If N is prime, then a^N = 1 (mod N) for all a < N.
If N is not prime, then a^N=1 (mod N) for at most half the values of a < N.
Thus, the probability of not getting a prime is ½ and the probability of getting a prime is 1 – 1/2. If you try k times then the probability become 1 – 1/(2^k).