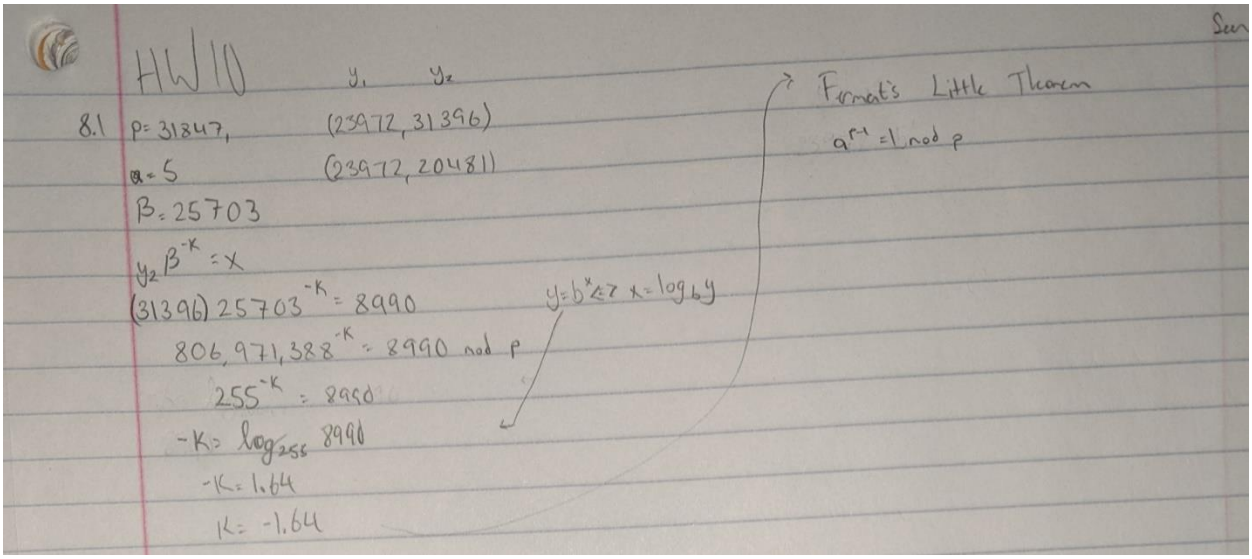
8.1



I tried to solve this by switching around some equations that I found on the PowerPoint, however none of them seemed to work. I wrote a program to brute force Fermat's Little Theorem, but due to some unseen logic error or maybe just the problem of having too large numbers, it would never function properly either. Even WolframAlpha aborted the calculation.

```
def fermats(p: int) -> int:
  pMinusOne = p - 1
  for i in range(2, p):
    if (i ** pMinusOne) % p == 1:
      return i

if __name__ == '__main__':
  p = 31847
  a = fermats(p)
  print(a)
```

8.2

I found pseudocode for this somewhere after searching for a long time, which I will append to the end. Apparently, it comes straight from built-in MATLAB functions. I then turned the pseudocode into workable Python code. The square and multiply function I have was taken from the V200's function list and converted to Python. **Part b** is found using a simple brute force algorithm to test all numbers in range of p against the formula from the PowerPoint: $\alpha^a = \beta(mod\ p)$. **Part c** is another brute force algorithm playing on the PowerPoint equation of $sam(\alpha, k, p) \rightarrow y_1$.

```
def sam(x, n, m):
  z = 1
  while n > 0:
    if n % 2 == 1:
      z = x*z % m
      n -= 1
    else:
      x = x**2 % m
      n /= 2
  return z

# Part A
def verify(message, alpha, beta, p, y):
  alpha_x = sam(alpha, message, p)
  beta_y1 = sam(beta, y[0], p)
  y1_y2 = sam(y[0], y[1], p)
  ver = (beta_y1 * y1_y2) % p

  if ver == alpha_x:
    return True
  else:
    return False

# Part B
def findPrivate(alpha, beta, p):
  for i in range(p):
    if (alpha ** i) % p == beta:
      return i

# Part C
def findK(alpha, p, y1):
  for k in range(p):
    calc = sam(alpha, k, p)
```

```
    if calc == y1:
      return k


if __name__ == '__main__':
  x = 20543
  p = 31847
  alpha = 5
  beta = 26379
  y = [20679, 11082]

  print(f'Part (a): {verify(x, alpha, beta, p, y)}')
  print(f'Part (b): {findPrivate(alpha, beta, p)}')
  print(f'Part (c): {findK(alpha, p, y[0])}')
```

Bonus Question:

To find this, I wrote a python program that would just check every number in the range of n to see if its square would equal one mod n.

```
def isOne(p:int, q:int) -> list:
  a = []
  n = p * q
  for i in range(n):
    if (i ** 2) % n == 1:
      a.append(i)
  return a

if __name__ == '__main__':
  p, q = 26981, 62549
  a = isOne(p, q)
  print(a)
```

[1,

35965674,

1651668895,

1687634568]

Pseudocode

%%Verification

alpha_x = square and multiply (alpha , message , p) ;

beta_gamma = square and multiply (beta , gamma, p);

gamma_delta = square and multiply (gamma, delta , p) ;

ver_aux = mod(beta gamma*gamma delta , p) ;

if (ver_aux == alpha_x)

verified = ' Verified ' ;

end