Trabalho Final

1. Como F_n é composto, para determinar se é pseudoprimo para a base 2 devemos calcular o resto de $2^{(F_n - 1)}$ por F_n.

Sabemos que $F_n \equiv 0 \pmod{F_n}$,

isto é, $2^{(2^n)} + 1 \equiv 0 \pmod{2^{(2^n)} + 1}$.

Portanto, obtemos que $2^{(2^n)} \equiv -1 \pmod{F(n)}$.

Notamos também que,

$F_n - 1 = 2^{(2^n)} = 2^{((2^n) - n)} . 2^n$.

Portanto,

$2^{(F_n - 1)} \equiv 2^{(2^{((2^n) - n)} . 2^n)} \equiv (2^{2^n})^{((2^n) - n)}$

$\equiv (-1)^{((2^n) - n)} \equiv 1 \pmod{F_n}$.

Logo, F_n é um pseudoprimo de Miller–Rabin para a base 2. ■

2. Por ser pseudoprimo de Miller-Rabin para a base b, sabemos que n é ímpar e é composto. Como n é ímpar, escrevemos

$n - 1 = (2^k).q$, onde $k \geq 1$ e q é um número ímpar.

Como n é pseudoprimo de Miller-Rabin para a base b, então

ou $b^q \equiv 1 \pmod{n}$ ou $b^{(2^j).q} \equiv -1 \pmod{n}$,

onde $0 \leq j \leq k - 1$.

No primeiro caso, temos que

$b^{(n - 1)} \equiv (b^q)^{(2^k)} \equiv (1)^{(2^k)} \equiv 1 \pmod{n}$.

E no segundo caso, temos

$b^{(n - 1)} \equiv (b^{(2^j).q})^{(2^{(k - j)})} \equiv (-1)^{(2^{(k - j)})}$.

Como $k > j$, logo $k - j \geq 1$ e, portanto,

$(-1)^{(2^{(k - j)})} \equiv 1 \pmod{n}$.

Em ambos os casos obtivemos que

$b^{(n - 1)} \equiv 1 \pmod{n}$.

Portanto, $b^n \equiv b \pmod{n}$.

Sabemos, pelo Teste de Primalidade de Fermat, que para n ser pseudoprimo de Fermat para a base b, n precisa ser composto e $b^n \equiv b \pmod{n}$.

Logo, n é um pseudoprimo de Fermat a base b. ■

3.

a.

```
[7] n = 8989
    fatoração_Fermat_v2(n)

    (89, 101)
```

Utilizando o algoritmo “fatoração_Fermat_v2” feito em aula(P 07), encontramos que $n = 8989 = 89 \cdot 101$
Com isso, $p = 89$, $q = 101$ e
$\Phi(n) = (p - 1)(q - 1) = 88.100 = 8800$.
Assim, devemos construir o menor expoente público “e” possível tal que exista “d” satisfazendo $ed \equiv 1 \mod 8800$.
Sabemos que para que “e” seja inversível mod 8800, é necessário que $mdc(e, 8800) = 1$.
Dessa forma, queremos o menor “e” > 1 ímpar tal que $mdc(e, 8800) = 1$.
Vamos tentar o menor ímpar > 1, ou seja, o número 3. Como 3 é primo e $8+8+0+0 = 16$ não é múltiplo de 3, isto é, 8800 não é múltiplo de 3, certamente, $mdc(3, 8800) = 1$.
Portanto, construímos $e = 3$.
Podemos utilizar o algoritmo “Euclides_estendido” feito em aula, para encontramos d.

```
[4] e = 3
    phi = 8800
    Euclides_estendido(e, phi)

    (1, -2933, 1)
```

Encontramos d = -2933.
Para encontrarmos a forma reduzida de d módulo 8800, efetuamos: -2933 + 8800 = 5867.
Logo, d = 5867.

b. Para codificar a mensagem 12345 vamos quebrá-la em dois blocos < n. (b_0 = 1234 e b_1 = 5).
Agora vamos encriptar, pelo python, cada bloco, fazendo exponenciação em módulo n = 8989 usando expoente e = 3.

```
[ ] def encripta_blocos(b0, b1):
      n = 8989
      e = 3
      c0 = pow(b0, e, n)
      c1 = pow(b1, e, n)
      return (c0, c1)

    encripta_blocos(1234, 5)

    (2366, 125)
```

Dessa forma, obtemos os blocos encriptados
c_0 = 2366, c_1 = 125.
Mensagem codificada: 2366, 125.

5. Porque, como p e q são primos e nos casos interessante são ímpares (para pq ser difícil de fatorar), $\varphi(n)$ é par. Assim, se e = 2 então, como $\varphi(n)$ é par, o mdc(e, $\varphi(n)$) será 2 e, com isso, o expoente “e” não será inversível e, portanto, não existirá seu inverso “d” satisfazendo $ed \equiv 1 \bmod \varphi(n)$.

8. Como sabemos que e = 4107 e $\varphi(n) = 6160$, podemos utilizar o algoritmo “Euclides_estendido” feito em aula, para encontramos d.

```
e = 4107
phi = 6160
Euclides_estendido(e, phi)
```

```
(1, 3, -2)
```

Encontramos d = 3.

Como conhecemos n = 6319, podemos utilizar a função "descriptar", feita na questão 12 desse trabalho, para decodificar a mensagem.

```
[80] blocos = [2823, 2688, 398, 4335, 2273]
     n = 6319
     d = 3
     descriptar(blocos, n, d)
```

```
'bicho'
```

Dessa forma, descobrimos que a mensagem é: “bicho”.