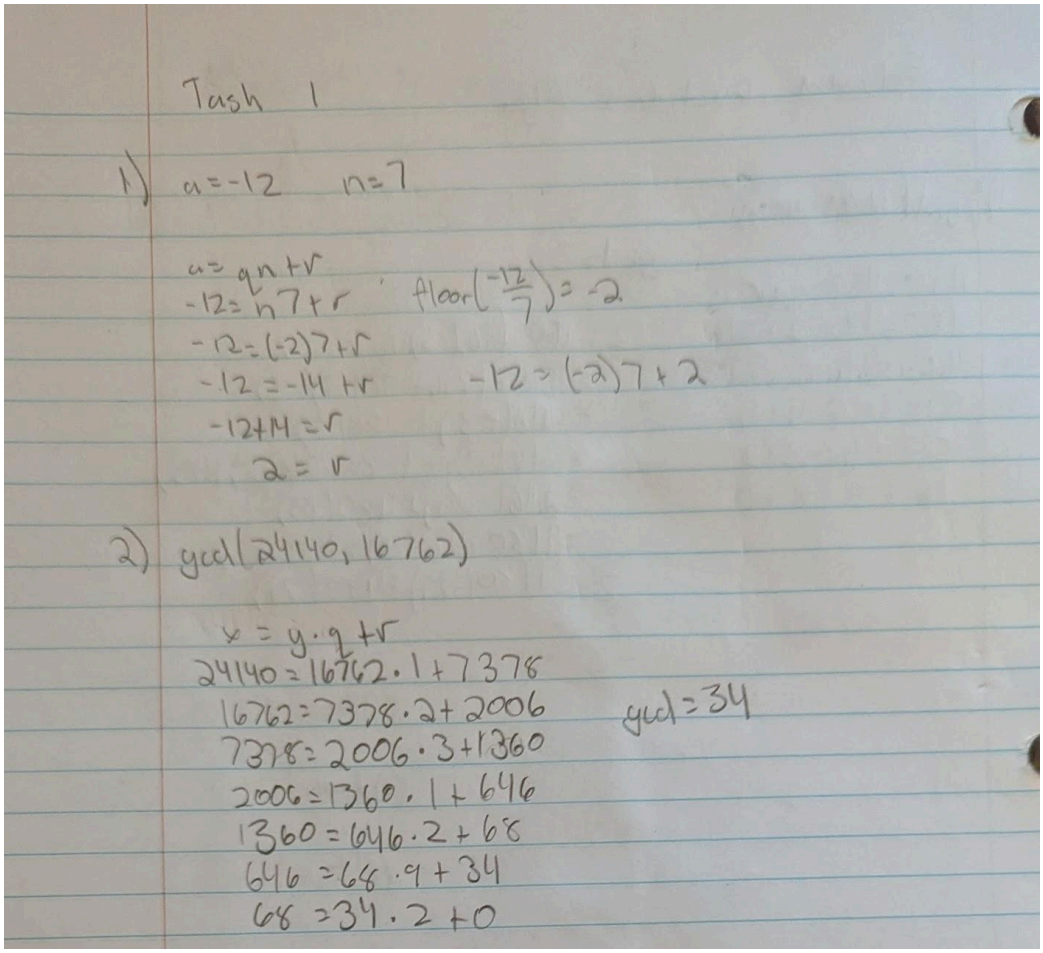
Assignment 1

COMP 7402

Task 1



Task 1

1) $a=-12 \quad n=7$

$a = qn + r$

$-12 = n7 + r$ $\quad \text{floor}\left(\frac{-12}{7}\right) = -2$

$-12 = (-2)7 + r$

$-12 = -14 + r \qquad -12 = (-2)7 + 2$

$-12 + 14 = r$

$2 = r$

2) $\gcd(24140, 16762)$

$x = y \cdot q + r$

$24140 = 16762 \cdot 1 + 7378$

$16762 = 7378 \cdot 2 + 2006 \qquad \gcd = 34$

$7378 = 2006 \cdot 3 + 1360$

$2006 = 1360 \cdot 1 + 646$

$1360 = 646 \cdot 2 + 68$

$646 = 68 \cdot 9 + 34$

$68 = 34 \cdot 2 + 0$

3) $5x = 4 \pmod 3$

$5 = 2 \pmod 3$

$4 = 1 \pmod 3$

$2x = 1 \pmod 3$

$2 \cdot 2 = 4 \equiv 1 \pmod 3$

$x = 2 \pmod 3$

4) $1234x = 1 \pmod{4321}$

$\gcd(4321, 1234)$

$4321 = 1234 \cdot 3 + 619$

$1234 = 614 \cdot 1 + 615$

$619 = 615 \cdot 1 + 4$

$615 = 4 \cdot 153 + 3$

$4 = 3 \cdot 1 + 1$

$3 = 1 \cdot 3 + 0$

$\gcd = 1$

$1 = 4 - 1 \cdot 3$

$1 = 4 - (615 - 153 \cdot 4)$

$1 = 154(619 - 615) - 615$

$1 = 154 \cdot 619 - 155(1234 - 619)$

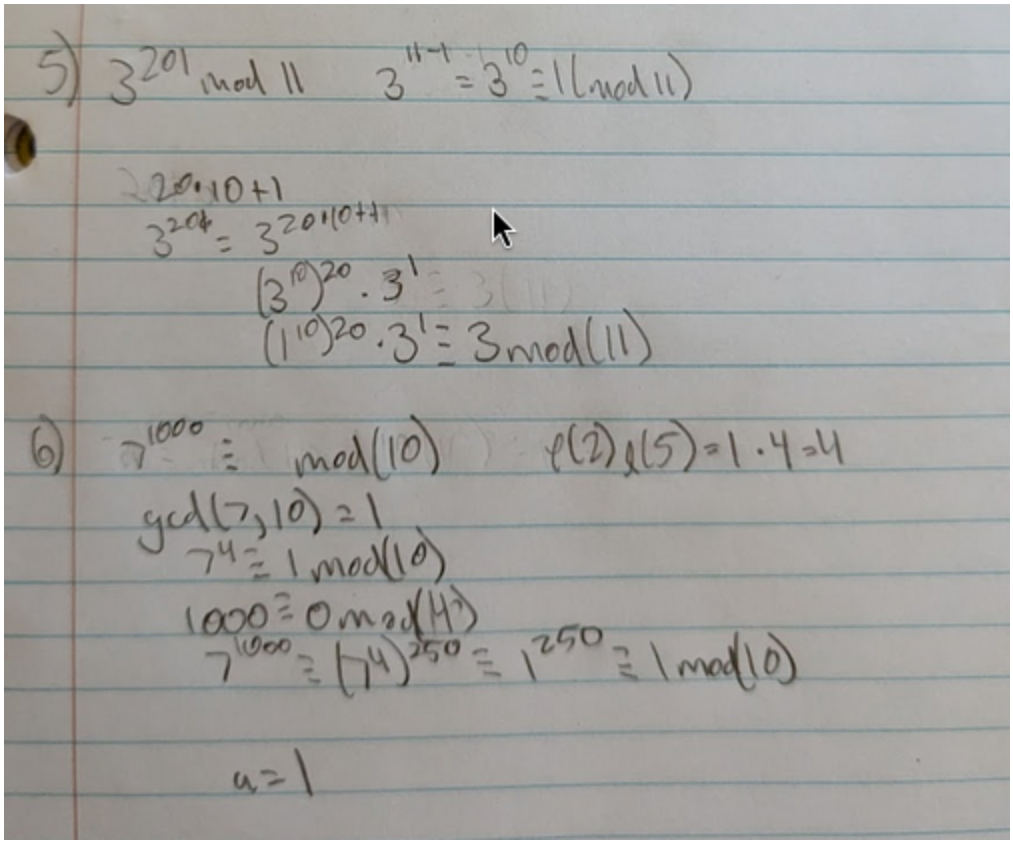
$1 = 309(4321 - 3 \cdot 1234) - 155 \cdot 1234$

$-1082 \cdot 1234 + 309 \cdot 4321 = 1$

$-1082 \equiv 4321 - 1082 = 3239 \pmod{4321}$

$1234^{-1} \equiv 3239 \pmod{4321}$

$inv = 3239$



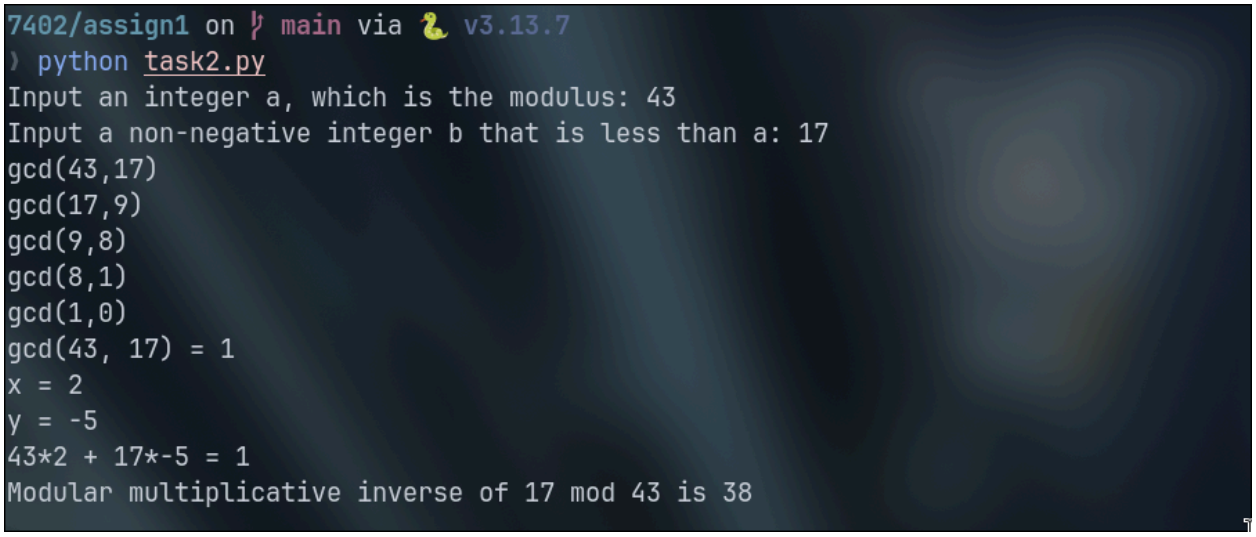
7. The miller rabin will run and return inconclusive if the number is thought to be prime, this is not exact. As the algorithm runs more the number has a higher chance to be a prime as it reduces the chances of it being a composite.

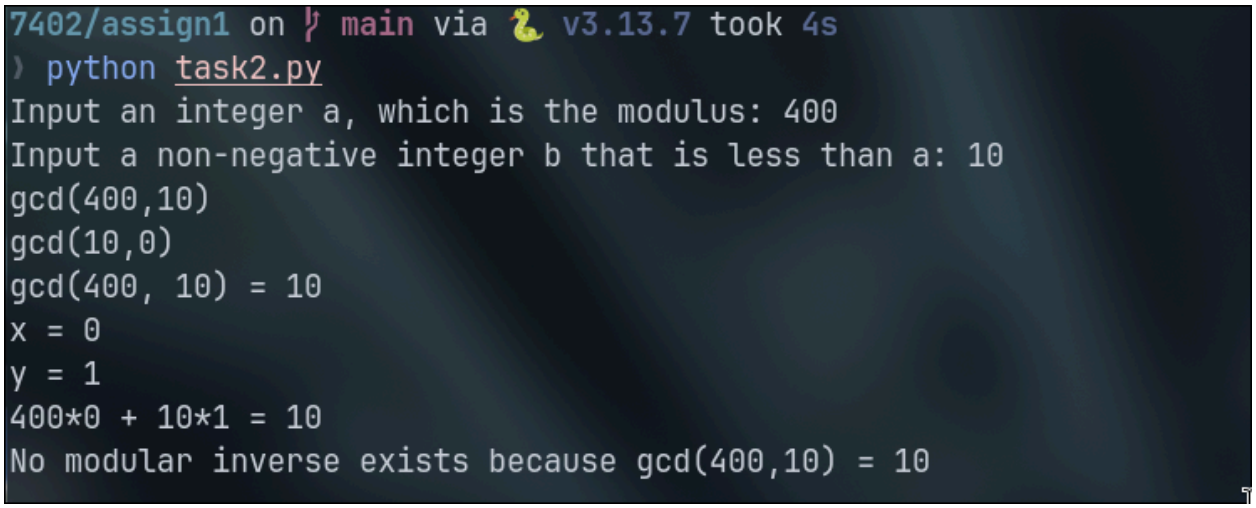
8)

i	$2^i \bmod 11$	a	$\log_{2,11} a$
1	2	2	1
2	4	4	2
3	8	8	3
4	5	5	4
5	10	10	5
6	9	9	6
7	7	7	7
8	3	3	8
9	6	6	9
10	1	1	10

Task 2

```python
def gcd(a, b):
    print(f"gcd({a},{b})")
    x0, x1 = 1, 0
    y0, y1 = 0, 1

    while b != 0:
        r = a % b
        q = a // b

        x_next = x0 - q * x1
        y_next = y0 - q * y1

        a, b = b, r
        x0, x1 = x1, x_next
        y0, y1 = y1, y_next

        print(f"gcd({a},{b})")

    return a, x0, y0

def main():
    a = int(input("Input an integer a, which is the modulus: "))
    b = int(input("Input a non-negative integer b that is less than a: "))
    gcd_val, x, y = gcd(a, b)

    print(f"gcd({a}, {b}) = {gcd_val}")
    print(f"x = {x}")
    print(f"y = {y}")
    print(f"{a}*{x} + {b}*{y} = {gcd_val}")

    if gcd_val == 1:
        mod_inv = y % a
        print(f"Modular multiplicative inverse of {b} mod {a} is {mod_inv}")
    else:
        print(f"No modular inverse exists because gcd({a},{b}) = {gcd_val}")

if __name__ == "__main__":
    main()
```


7402/assign1 on main via v3.13.7
python task2.py
Input an integer a, which is the modulus: 43
Input a non-negative integer b that is less than a: 17
gcd(43,17)
gcd(17,9)
gcd(9,8)
gcd(8,1)
gcd(1,0)
gcd(43, 17) = 1
x = 2
y = -5
43*2 + 17*-5 = 1
Modular multiplicative inverse of 17 mod 43 is 38


7402/assign1 on main via v3.13.7 took 4s
python task2.py
Input an integer a, which is the modulus: 400
Input a non-negative integer b that is less than a: 10
gcd(400,10)
gcd(10,0)
gcd(400, 10) = 10
x = 0
y = 1
400*0 + 10*1 = 10
No modular inverse exists because gcd(400,10) = 10

Task 3

```python
def create_matrix(keyword):
    #make a 5x5 matrix
    key = keyword.upper().replace(" ", "").replace("J", "I")

    check = set()
    unique_key = []
    for char in key:
        if char not in check:
            unique_key.append(char)
            check.add(char)

    letters = "ABCDEFGHIKLMNOPQRSTUVWXYZ"

    matrix_chars = unique_key
    for char in letters:
        if char not in check:
            matrix_chars.append(char)
            check.add(char)

    playfair_matrix = []
    for i in range(0, 25, 5):
        playfair_matrix.append(matrix_chars[i:i+5])

    return playfair_matrix

def display_playfair(matrix):
    print("Playfair 5x5 Matrix:")
    print("+" + "-" * 11 + "+")
    for row in matrix:
        print("| " + " ".join(row) + " |")
    print("+" + "-" * 11 + "+")

def search_pos_of_char(matrix, char):
    for i, row in enumerate(matrix):
        if char in row:
            return i, row.index(char)
    return None
```

```python
def format_text(plaintext):
    text = ''.join(c.upper() for c in plaintext if c.isalpha()).replace('J', 'I')
    result = []
    i = 0
    while i < len(text):
        a = text[i]
        if i + 1 < len(text):
            b = text[i + 1]
            if a == b:
                result.append(a + 'X')
                i += 1
            else:
                result.append(a + b)
                i += 2
        else:
            result.append(a + 'X')
            i += 1
    return result

def encrypt_digraph(matrix, digraph):
    (r1, c1) = search_pos_of_char(matrix, digraph[0])
    (r2, c2) = search_pos_of_char(matrix, digraph[1])

    if r1 == r2:
        return matrix[r1][(c1+1)%5] + matrix[r2][(c2+1)%5]
    elif c1 == c2:
        return matrix[(r1+1)%5][c1] + matrix[(r2+1)%5][c2]
    else:
        return matrix[r1][c2] + matrix[r2][c1]

def encrypt_text(matrix, plaintext):
    digraphs = format_text(plaintext)
    ciphertext = ''.join(encrypt_digraph(matrix, dg) for dg in digraphs)
    return ciphertext

def main():
    keyword = input("Enter keyword: ").strip()
    if not keyword:
        print("Error: No keyword entered!")
        return

    plaintext = input("Enter plaintext: ").strip()
    if not plaintext:
        print("Error: No plaintext entered!")
        return
```

```
84
85      pf_matrix = create_matrix(keyword)
86      display_playfair(pf_matrix)
87
88      ciphertext = encrypt_text(pf_matrix, plaintext)
89      print(f"\nPlaintext : {plaintext}")
90      print(f"Ciphertext: {ciphertext}")
91
92  if __name__ == "__main__":
93      main()
94
```

```
7402/assign1 on  main [!] via  v3.13.7
❯ python task3.py
Enter keyword: MONARCHY
Enter plaintext: test
Playfair 5x5 Matrix:
+-----------+
| M O N A R |
| C H Y B D |
| E F G I K |
| L P Q S T |
| U V W X Z |
+-----------+

Plaintext : test
Ciphertext: LKTL
```

```
7402/assign1 on  main [!] via  v3.13.7 took 8s
❯ python task3.py
Enter keyword: BALLOON
Enter plaintext: banana
Playfair 5x5 Matrix:
+-----------+
| B A L O N |
| C D E F G |
| H I K M P |
| Q R S T U |
| V W X Y Z |
+-----------+

Plaintext : banana
Ciphertext: ALBLBL
```

```
7402/assign1 on  main [!] via  v3.13.7 took 11s
❯ python task3.py
Enter keyword: TEST THREE
Enter plaintext: playfair cipher
Playfair 5x5 Matrix:
+-----------+
| T E S H R |
| A B C D F |
| G I K L M |
| N O P Q U |
| V W X Y Z |
+-----------+

Plaintext : playfair cipher
Ciphertext: QKDVABMEBKQSST
```