II Sem. M.Tech(Software Engineering)

ICT 5028 Cyber Security

Assignment -1

Submitted By:

Archana Vikraman Nair

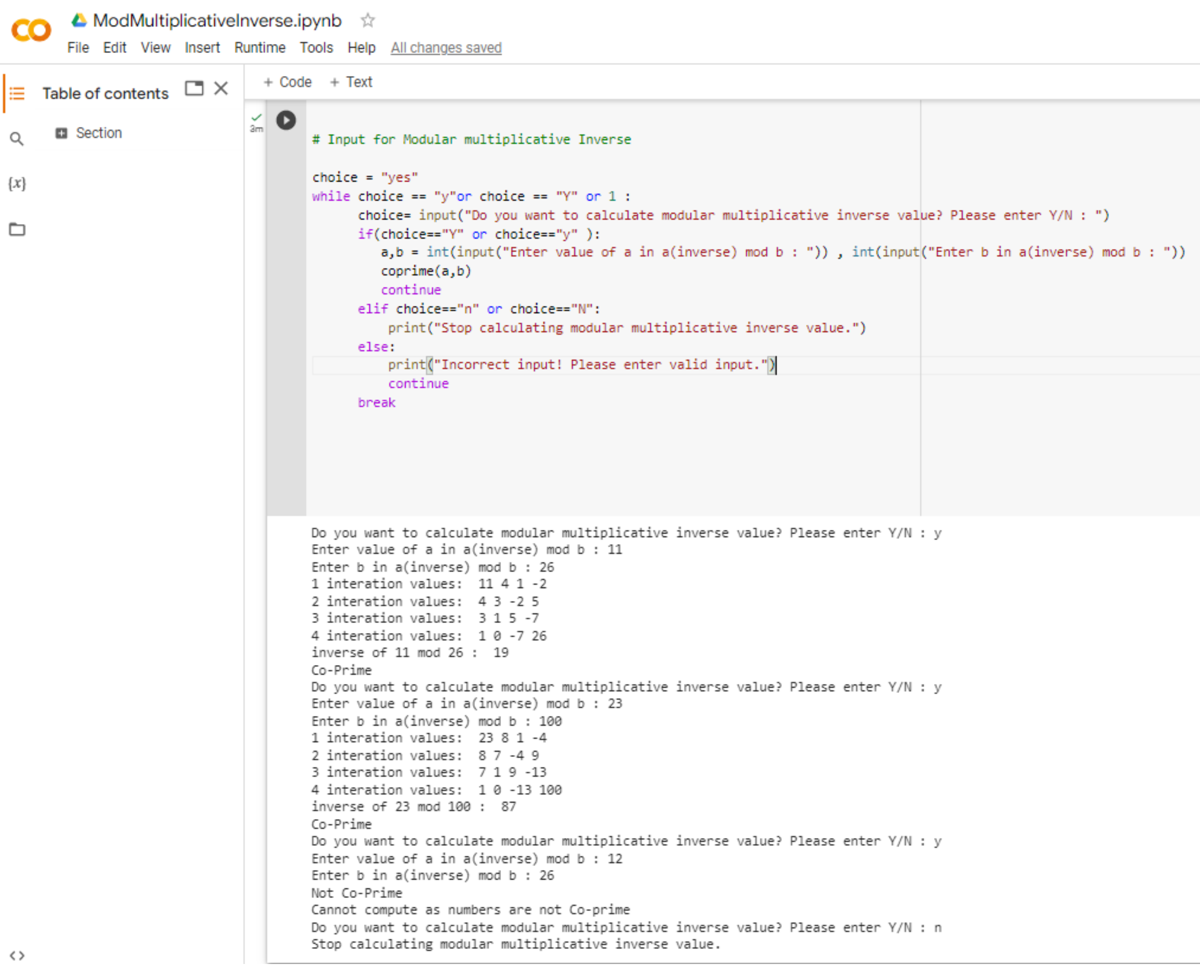
Reg No: 220928003

1) Modular multiplicative inverse of a number using extended Euclidian Algorithm

Verify following using program:

i. 11^-1 mod 26 = 19
ii. 23^-1 mod 100 = 87
iii. 12^-1 mod 26 = The gcd (26, 12) = 2 ≠ 1, which means there is no multiplicative inverse for 12 in Z 26.

Example 1, 2 and 3: outputs




```
# Input for Modular multiplicative Inverse

choice = "yes"
while choice == "y"or choice == "Y" or 1 :
      choice= input("Do you want to calculate modular multiplicative inverse value? Please enter Y/N : ")
      if(choice=="Y" or choice=="y" ):
         a,b = int(input("Enter value of a in a(inverse) mod b : ")) , int(input("Enter b in a(inverse) mod b : "))
         coprime(a,b)
         continue
      elif choice=="n" or choice=="N":
          print("Stop calculating modular multiplicative inverse value.")
      else:
          print("Incorrect input! Please enter valid input.")
          continue
      break
```

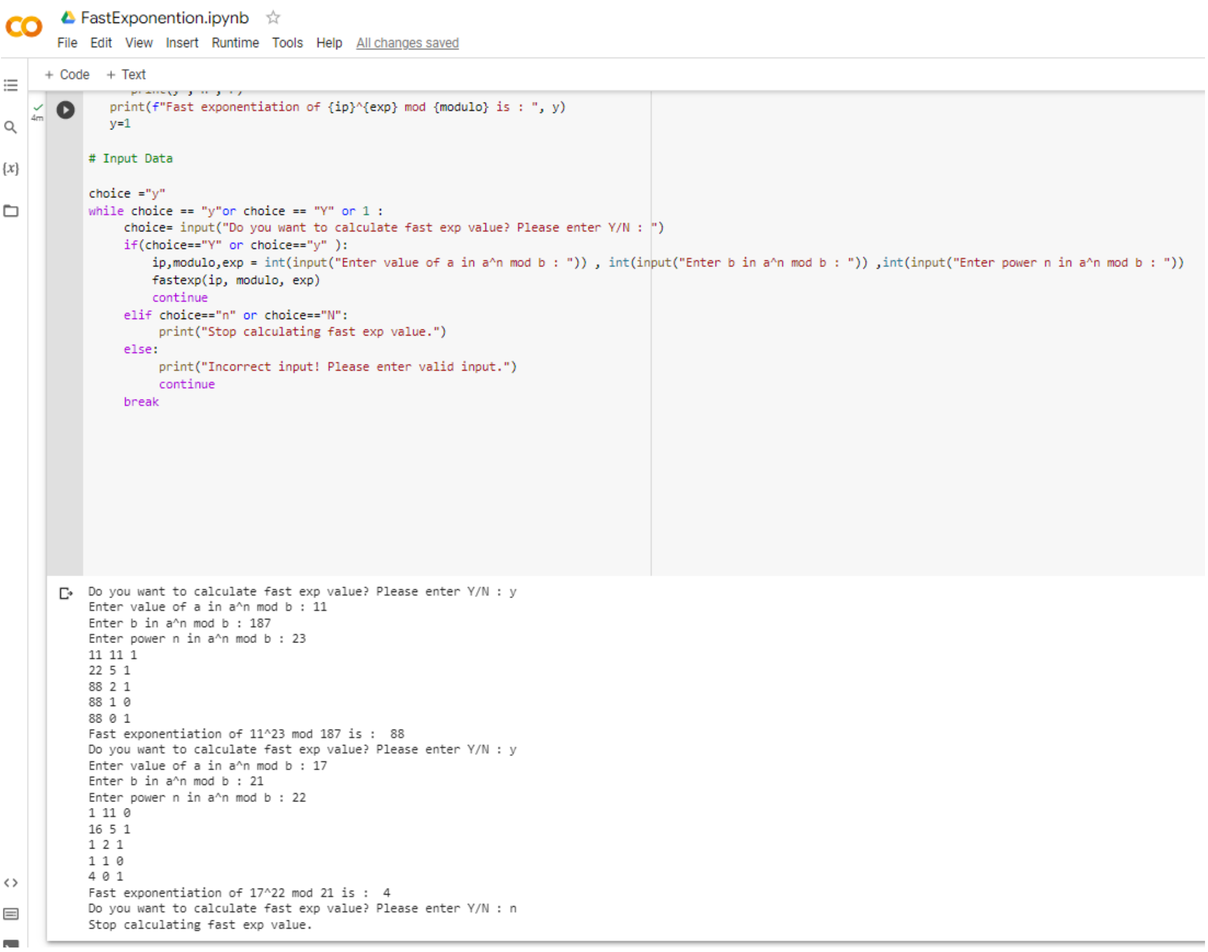
```
Do you want to calculate modular multiplicative inverse value? Please enter Y/N : y
Enter value of a in a(inverse) mod b : 11
Enter b in a(inverse) mod b : 26
1 interation values:  11 4 1 -2
2 interation values:  4 3 -2 5
3 interation values:  3 1 5 -7
4 interation values:  1 0 -7 26
inverse of 11 mod 26 :  19
Co-Prime
Do you want to calculate modular multiplicative inverse value? Please enter Y/N : y
Enter value of a in a(inverse) mod b : 23
Enter b in a(inverse) mod b : 100
1 interation values:  23 8 1 -4
2 interation values:  8 7 -4 9
3 interation values:  7 1 9 -13
4 interation values:  1 0 -13 100
inverse of 23 mod 100 :  87
Co-Prime
Do you want to calculate modular multiplicative inverse value? Please enter Y/N : y
Enter value of a in a(inverse) mod b : 12
Enter b in a(inverse) mod b : 26
Not Co-Prime
Cannot compute as numbers are not Co-prime
Do you want to calculate modular multiplicative inverse value? Please enter Y/N : n
Stop calculating modular multiplicative inverse value.
```

2) Fast exponentiation in congruences

Verify following using program:

i. 11^23 mod 187 = 88
ii. 17^22 mod 21 = 4

Example 1 and 2 ouputs:


FastExponention.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
    print(f"Fast exponentiation of {ip}^{exp} mod {modulo} is : ", y)
    y=1

# Input Data

choice ="y"
while choice == "y"or choice == "Y" or 1 :
     choice= input("Do you want to calculate fast exp value? Please enter Y/N : ")
     if(choice=="Y" or choice=="y" ):
         ip,modulo,exp = int(input("Enter value of a in a^n mod b : ")) , int(input("Enter b in a^n mod b : ")) ,int(input("Enter power n in a^n mod b : "))
         fastexp(ip, modulo, exp)
         continue
     elif choice=="n" or choice=="N":
          print("Stop calculating fast exp value.")
     else:
          print("Incorrect input! Please enter valid input.")
          continue
     break
```

```
Do you want to calculate fast exp value? Please enter Y/N : y
Enter value of a in a^n mod b : 11
Enter b in a^n mod b : 187
Enter power n in a^n mod b : 23
11 11 1
22 5 1
88 2 1
88 1 0
88 0 1
Fast exponentiation of 11^23 mod 187 is :  88
Do you want to calculate fast exp value? Please enter Y/N : y
Enter value of a in a^n mod b : 17
Enter b in a^n mod b : 21
Enter power n in a^n mod b : 22
1 11 0
16 5 1
1 2 1
1 1 0
4 0 1
Fast exponentiation of 17^22 mod 21 is :  4
Do you want to calculate fast exp value? Please enter Y/N : n
Stop calculating fast exp value.
```

3) Public key cryptosystems

Diffie Hellman Key exchange Algorithm

Verify following two examples using program:

Example 1: Let g = 7 and p = 23.

The steps are as follows:

1. Alice chooses x = 3 and calculates R1 = 7^3 mod 23 = 21.
2. Bob chooses y = 6 and calculates R2 = 7^6 mod 23 = 4.
3. Alice sends the number 21 to Bob.

4. Bob sends the number 4 to Alice.
5. Alice calculates the symmetric key K = 4^3 mod 23 = 18.
6. Bob calculates the symmetric key K = 21^6 mod 23 = 18.
The value of K is the same for both Alice and Bob; g^xy mod p = 7^18 mod 23 = 18.

Diffie-Hellman.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```python
  if primitive_check(G, P, 1) == -1:
    print(f"Number Is Not A Primitive Root Of {P}, Please Try Again!")
    continue
  break

# Private Keys
x1, x2 = int(input("Enter The Private Key Of User 1 : ")), int(
  input("Enter The Private Key Of User 2 : "))
while 1:
  if x1 >= P or x2 >= P:
    print(f"Private Key Of Both The Users Should Be Less Than {P}!")
    continue
  break

# Calculate Public Keys
y1, y2 = pow(G, x1) % P, pow(G, x2) % P

# Generate Secret Keys
k1, k2 = pow(y2, x1) % P, pow(y1, x2) % P

print(f"\nSecret Key For User 1 Is {k1}\nSecret Key For User 2 Is {k2}\n")

if k1 == k2:
  print("Keys Have Been Exchanged Successfully")
else:
  print("Keys Have Not Been Exchanged Successfully")
```

```
Enter P : 23
Enter The Primitive Root Of 23 : 7
Enter The Private Key Of User 1 : 3
Enter The Private Key Of User 2 : 6

Secret Key For User 1 Is 18
Secret Key For User 2 Is 18

Keys Have Been Exchanged Successfully
```

Example 2: Alice and Bob have chosen prime value q = 17 and primitive root = 5. If Alice's secret key is 4 and Bob's secret key is 6.

Both Alice and Bob calculate the value of their public key and exchange with each other.

Public key of Alice

- = 5 private key of Alice mod 17
- = 5^4 mod 17
- = 13

Public key of Bob

- = 5 private key of Bob mod 17
- = 5^6 mod 17
- = 2

Both the parties calculate the value of secret key at their respective side.

Secret key obtained by Alice

- = 2 private key of Alice mod 7
- = 2^4 mod 17
- = 16

Secret key obtained by Bob

- = 13 private key of Bob mod 7
- = 13^6 mod 17
- = 16

Finally, both the parties obtain the same value of secret key.

The value of common secret key = 16.

Diffie-Hellman.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```python
while 1:
  G = int(input(f"Enter The Primitive Root Of {P} : "))
  if primitive_check(G, P, 1) == -1:
    print(f"Number Is Not A Primitive Root Of {P}, Please Try Again!")
    continue
  break

# Private Keys
x1, x2 = int(input("Enter The Private Key Of User 1 : ")), int(
  input("Enter The Private Key Of User 2 : "))
while 1:
  if x1 >= P or x2 >= P:
    print(f"Private Key Of Both The Users Should Be Less Than {P}!")
    continue
  break

# Calculate Public Keys
y1, y2 = pow(G, x1) % P, pow(G, x2) % P

# Generate Secret Keys
k1, k2 = pow(y2, x1) % P, pow(y1, x2) % P

print(f"\nSecret Key For User 1 Is {k1}\nSecret Key For User 2 Is {k2}\n")

if k1 == k2:
  print("Keys Have Been Exchanged Successfully")
else:
  print("Keys Have Not Been Exchanged Successfully")
```

```
Enter P : 17
Enter The Primitive Root Of 17 : 5
Enter The Private Key Of User 1 : 4
Enter The Private Key Of User 2 : 6

Secret Key For User 1 Is 16
Secret Key For User 2 Is 16

Keys Have Been Exchanged Successfully
```

RSA Algorithm

Verify following two examples using program:

Example 1: Bob chooses 7 and 11 as p and q and calculates n = 7 × 11 = 77. The value of ϕ(n) = (7 − 1) *(11 − 1) or 60. Now he chooses two exponents, e and d, from Z60∗. If he chooses e to be 13, then d is 37.

Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5.

Plaintext: 5	$C = 5^{13} = 26 \bmod 77$	Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26	$P = 26^{37} = 5 \bmod 77$	Plaintext: 5

The plaintext 5 sent by Alice is received as plaintext 5 by Bob.

RSA-method.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

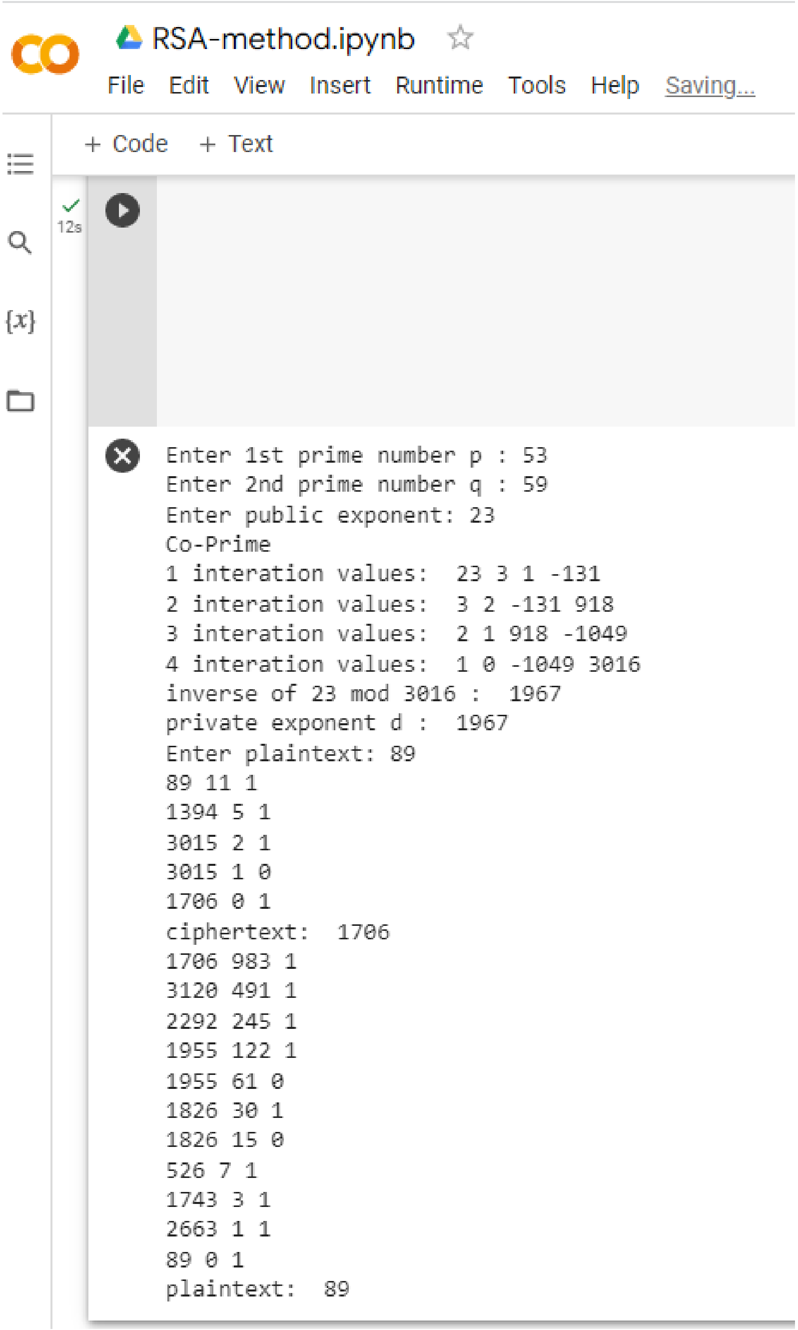
```
if private_exponent(p,q,e)==-1 :
   print("Please enter correct value of e")
   continue
break
```

```
Enter 1st prime number p : 7
Enter 2nd prime number q : 11
Enter public exponent: 13
Co-Prime
1 interation values:  13 8 1 -4
2 interation values:  8 5 -4 5
3 interation values:  5 3 5 -9
4 interation values:  3 2 -9 14
5 interation values:  2 1 14 -23
6 interation values:  1 0 -23 60
inverse of 13 mod 60 :  37
private exponent d :  37
Enter plaintext: 5
5 6 1
5 3 0
45 1 1
26 0 1
ciphertext:  26
26 18 1
26 9 0
45 4 1
45 2 0
45 1 0
5 0 1
plaintext:  5
```

Example 2: Select two prime no's. Suppose P = 53 and Q = 59. Now first part of the public key: n = P*Q = 3127.

Now we are ready with our – Public Key (n = 3127 and e = 23) and Private Key (d = 1967)
Now we will encrypt **"HI"**: Convert letters to numbers: H = 8 and I = 9.
Thus, **Encrypted Data c = $89^{e \bmod n}$.**
Thus, Encrypted Data comes out to be 1706.

Now we will decrypt **1706**: **Decrypted Data = $c^{d \bmod n}$.**
Thus, our Encrypted Data comes out to be 89. **8 = H and I = 9 i.e., "HI".**



Symmetric Key Cryptosystems

4) Simplified DES

Output 1 for plaintext: 654 and Key value = 732. Ciphertext: 1b 71 fe and plaintext received is 654.

simplified_des ArchanNair > main.py > ...

```python
# Author Details :- Archana Nair, Manipal Institute of Technology, Reg No. 220928003
from SDES import SDES
import time

if __name__ == "__main__":

    # Input the plaintext
    print("Enter the plaintext ; ")
    plaintext = input().rstrip()
    # Input the key
    print("Enter the key. Value should be in range 0 to {} as the key size is {}".format(2 ** SDES.key_size - 1, SDES.key_size))
    key = int(input())
    if (key < 0) or (key > (2 ** SDES.key_size - 1)) :
        print("Follow the rules for the key")
        exit(1)

    print("Length of Text : ", len(plaintext))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\sesa627346\Downloads\CyberSecurity_LABAssignment 1 Archana Nair> & C:/Users/sesa627346/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/sesa627346/Downloads/CyberSecurity_LABAssignment 1 Archana Nair/simplified_des ArchanNair/main.py"
Enter the plaintext ;
654
Enter the key. Value should be in range 0 to 1023 as the key size is 10
732
Length of Text :  3
Encrypting...
Ciphertext :  1b 71 fe
Time required for encryption : 0.0
000110110111000111111110
Decrypting...
Plaintext :  654
Time required for decryption : 0.0
PS C:\Users\sesa627346\Downloads\CyberSecurity_LABAssignment 1 Archana Nair>
```

Output 2 for plaintext: 8312 and Key value = 487. Ciphertext: ee ff 31 2e and plaintext received is 8312.

simplified_des ArchanNair > main.py > ...

```python
# Author Details :- Archana Nair, Manipal Institute of Technology, Reg No. 220928003
from SDES import SDES
import time

if __name__ == "__main__":

    # Input the plaintext
    print("Enter the plaintext ; ")
    plaintext = input().rstrip()
    # Input the key
    print("Enter the key. Value should be in range 0 to {} as the key size is {}".format(2 ** SDES.key_size - 1, SDES.key_size
    key = int(input())
    if (key < 0) or (key > (2 ** SDES.key_size - 1)) :
        print("Follow the rules for the key")
        exit(1)

    print("Length of Text : ", len(plaintext))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\sesa627346\Downloads\CyberSecurity_LABAssignment 1 Archana Nair> & C:/Users/sesa627346/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/sesa627346/Downloads/CyberSecurity_LABAssignment 1 Archana Nair/simplified_des ArchanNair/main.py"
Enter the plaintext ;
8312
Enter the key. Value should be in range 0 to 1023 as the key size is 10
487
Length of Text :  4
Encrypting...
Ciphertext :  eb ff 31 2e
Time required for encryption : 0.0
11101011111111110011000100101110
Decrypting...
Plaintext :  8312
Time required for decryption : 0.0
PS C:\Users\sesa627346\Downloads\CyberSecurity_LABAssignment 1 Archana Nair>
```