* To implement Diffie Hellmens exchange, both parties A & B privately, mutually agree on positive number p & q.

$p \Rightarrow$ prime number

$q \rightarrow$ generator of p (primitive root)

A generator is a number such that when raised to a positive whole number power less than p, it never produces the same result.

$(q < p)$

A	B
Public key = P, G	Public Key = P, G
Private key = a	Private key = b
Key = $x = G^{a} \bmod P$	Key = $y = G^{b} \bmod P$

! Exchange of keys !

Key = y	Key = x
Secret Key K_a = $y^{a} \bmod P$	Secret Key k_b = $x^{b} \bmod P$

$$\boxed{K_a = K_b}$$

$q = 17$ $\quad\quad$ $a = 4$

$n = 5$ $\quad\quad$ $b = 6$

Publickey (A) $= 3^4 \bmod 17$

$= 13$

Public key (B) $= 5^6 \bmod 17$

$= 2$

Secret key (A) $= 2^4 \bmod 17$

$= 16$

Secret key (B) $= 13^8 \bmod 17$

$= 16$

$\therefore$ ⓐ $\underline{\underline{16}}$

7) Encryption: string = "Aaron Mendonca"
                  Keyword = "HELLO"

```
        def generatekey (string, key):
                    key = list (key)
                    if len (string) = len (key):
                        return (key)
                    else:
                        for i in range ((len (String) - len (key)
                            key.append (key [i % len (key
                    return (" ".join (key))
        def enclypt_cipher (string, key):
                cipher_text = [ ]
                  for i in range (len (string)):
                x = ((ord (string [i]) + ord (key [i] % 26
                ciphertext.append (chr (x))                tord
                return (" ".join (ciphertext))

key = generateKey (string, keyword)
cipher_text = enclypt_cipher (string, key)

  Print (" Plain text ", string)
  print ("Keyword:" keyword)
  print (" Cipher ", ciphertext).
```

```
string= "GEEKSFORGEEKS"
keyword= "AYUSH"
def generatekey(string,key):
    Key : list(key)
    if len(string) : = len(key):
        return(key)
    else:
        for i in range(len(string)-len(key)):
        Key.append(Key[i % len(key)])
    return("".join(key))
def ~~ciphertext~~ original (~~string~~ ciphertext, key) :
    orig_text = []
    for i in range(len(ciphertext)):
        x = (ord(ciphertext[i]) - ord(key[i]) + 26) % 26
        x += ord('A')
        orig_text.append(chr(x))
    return("".join(orig_text)
key = generatekey(string, keyword)
print("Original text :", originaltext (string, key)
```