**Ex. No.:2B**

**Date: 02/03/2024**

**DIFFIE HELLMANN ALGORITHM**

**Aim:**

To write a Python program to implement the Diffie-Hellmann algorithm.

**Algorithm:**

1. P,G => available public keys.
2. a is selected as a private key.

b is selected as a private key.

1. Equation to generate key: x=G mod p.

Equation to generate key: y=G mod p

1. After exchanging keys, user1 receives key y. After exchanging keys, user2 receives key x.

**Program:**

**#Diffie Hellmann Algorithm**

import math

def prime\_checker(p):

# Checks if the number entered is Prime or not

if p < 1:

return False

elif p > 1:

if p == 2:

return True

for i in range(2, int(math.sqrt(p)) + 1):

if p % i == 0:

return False

return True

def primitive\_check(g, p, L):

# Checks if the entered number is a primitive root or not

for i in range(1, p):

L.append(pow(g, i) % p)

for i in range(1, p):

if L.count(i) > 1:

L.clear()

return False

return True

l = []

while True:

P = int(input("Enter P: "))

if not prime\_checker(P):

print("No is not prime please try again!")

continue

break

while True:

G = int(input(f"Enter the primitive root of {P}: "))

if not primitive\_check(G, P, l):

print(f"Number is not a primitive root of {P}, please try again!")

continue

break

# Private Keys

x1 = int(input("Enter the private key of user 1: "))

x2 = int(input("Enter the private key of user 2: "))

while True:

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 = pow(G, x1) % P

y2 = pow(G, x2) % P

# Generate Secret Keys

k1 = pow(y2, x1) % P

k2 = pow(y1, x2) % P

print(f"\nSecret Key for user 1 is {k1}\nSecret key for user 2 is {k2}\n")

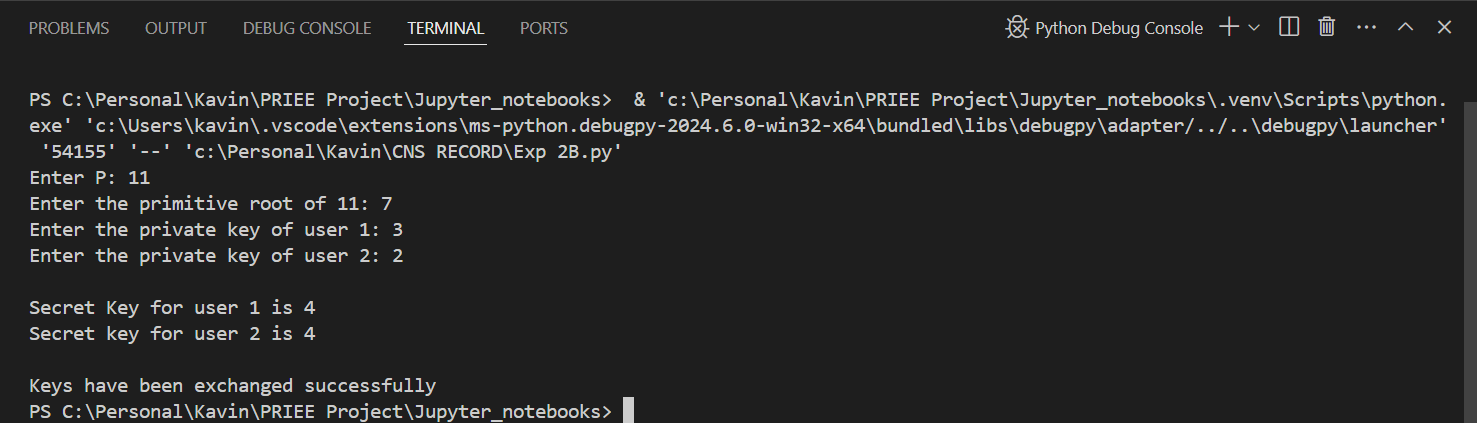
if k1 == k2:

print("Keys have been exchanged successfully")

else:

print("Keys have not been exchanged successfully")

**Output:**



**Result:**

Hence, Diffie Hellmann Algorithm has been implemented successfully.