**Exp: 2B** **Diffie Hellman Algorithm**
**Date: 16-03-2024**

**Aim:**

To write a python program implementing the Diffie Hellman algorithm.

**Algorithm:**

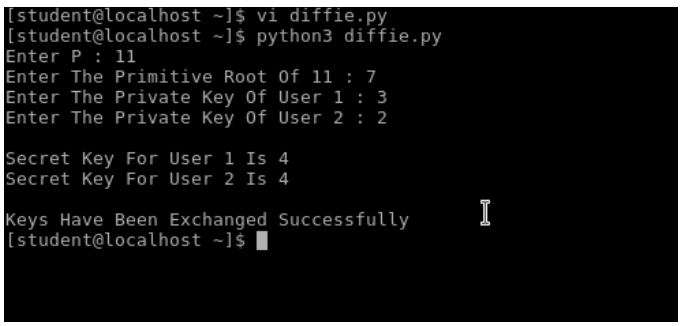
1. P, G => available public keys. P, G => available public keys.
2. a is selected as a private key. b is selected as a private key.
3. Eq. to generate key: $x=G^a$ modP. Eq. to generate key: $y=G^b$ modP.
4. After exchanging keys, user1 receives key y. After exchanging keys, user2 receives key x.

**Program:**

```
def prime_checker(p):
      if p < 1:
       return -1
   elif p > 1:
      if p == 2:
         return 1
      for i in range(2, p):
         if p % i == 0:
            return -1
         return 1

def primitive_check(g, p, L):
      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 -1
      return 1


l = []
```

```
while 1:
    P = int(input("Enter P : "))
    if prime_checker(P) == -1:
        print("Number Is Not Prime, Please Enter Again!")
        continue
    break
while 1:
    G = int(input(f"Enter The Primitive Root Of {P} : "))
    if primitive_check(G, P, l) == -1:
        print(f"Number Is Not A Primitive Root Of {P}, Please Try Again!")
        continue
    break
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
y1, y2 = pow(G, x1) % P, pow(G, x2) % P
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")
```

**Output:**



**Result:**

Thus the python program for the Diffie Hellman algorithm is implemented successfully.