**Experiment 4**

**Lavanya Bhambure MITU20BTCS0150 LY ISA 1 2203083**

**Title:**

Diffie Hellman Algorithm

**Problem Statement:**

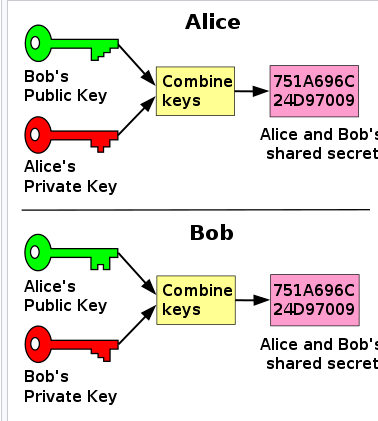
Implement Diffie Hellman key exchange algorithm for secret key generation and distribution of public key.

**Aim:**

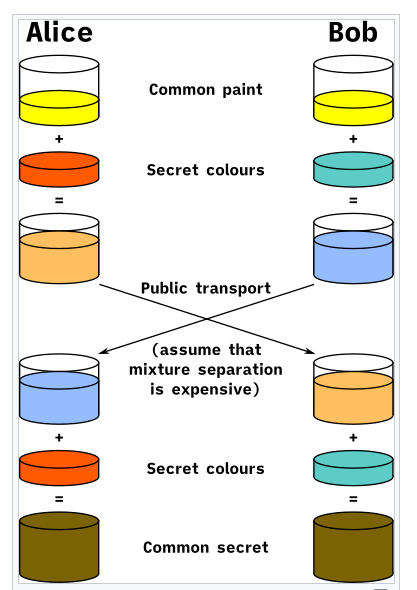
To understand and implement Diffie Hellman Algorithm

**Theory:**

The Diffie-Hellman key exchange algorithm is a fundamental cryptographic protocol that allows two parties to securely establish a shared secret key over an insecure communication channel. This shared key can then be used for secure communication, such as encrypting and decrypting messages.

****

The process begins by having the two parties, [Alice and Bob](https://en.wikipedia.org/wiki/Alice_and_Bob), publicly agree on an arbitrary starting colour that does not need to be kept secret. In this example, the colour is yellow. Each person also selects a secret colour that they keep to themselves – in this case, red and cyan. The crucial part of the process is that Alice and Bob each mix their own secret colour together with their mutually shared colour, resulting in orange-tan and light-blue mixtures respectively, and then publicly exchange the two mixed colours. Finally, each of them mixes the colour they received from the partner with their own private colour. The result is a final colour mixture (yellow-brown in this case) that is identical to their partner's final colour mixture.

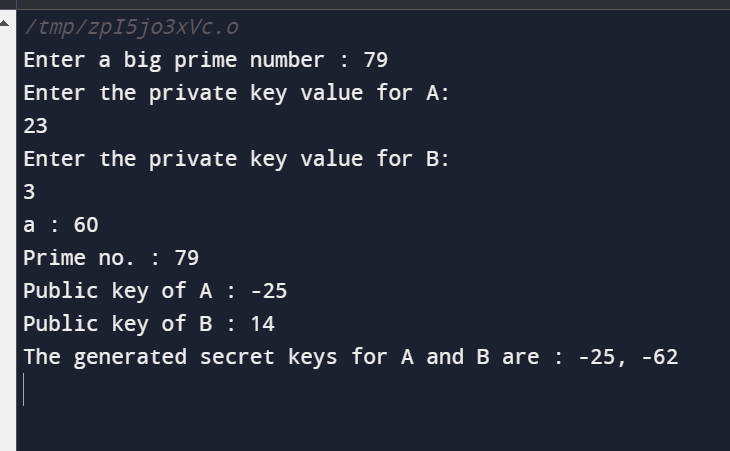


If a third party listened to the exchange, they would only know the common colour (yellow) and the first mixed colours (orange-tan and light-blue), but it would be very hard for them to find out the final secret colour (yellow-brown). Bringing the analogy back to a [real-life](https://en.wikipedia.org/wiki/Real-life) exchange using large numbers rather than colours, this determination is computationally expensive. It is impossible to compute in a practical amount of time even for modern [supercomputers](https://en.wikipedia.org/wiki/Supercomputer).

**Algorithm:**

* #include <iostream>
* #include <unordered\_map>
* #include <math.h>
* using namespace std;
* class Diffie\_Hellman
* {
* private:
* int p, a, Xa, Xb, Ya, Yb, ka, kb;
* int k; // k is a temp variable
* public:
* Diffie\_Hellman()
* {
* cout << "Enter a big prime number : ";
* cin >> p;
* cout << endl;
* }
* void getting\_a()
* {
* unordered\_map<int, int> primitive\_root;
* for (a = 1; a < p; a++)
* {
* for (int i = 1; i < p - 1; i++)
* {
* int k = pow(a, i);
* primitive\_root[k % p] = i;
* }
* if (primitive\_root.size() == (p - 1))
* {
* break;
* }
* }
* }
* void getPrivateKeys()
* {
* cout << "Enter the private key value for A: " << endl;
* cin >> Xa;
* cout << "Enter the private key value for B: " << endl;
* cin >> Xb;
* }
* void calculatePublicKeys()
* {
* k = (int)pow(a, Xa);
* Ya = k % p;
* k = (int)pow(a, Xb);
* Yb = k % p;
* }
* void generated\_SecretKey()
* {
* k = (int)pow(Yb, Xa);
* ka = k % p;
* k = (int)pow(Ya, Xb);
* kb = k % p;
* }
* void print()
* {
* cout << "a : " << a << endl;
* cout << "Prime no. : " << p << endl;
* cout << "Public key of A : " << Ya << endl;
* cout << "Public key of B : " << Yb << endl;
* cout<<"The generated secret keys for A and B are : "<<ka<<", "<<kb<<endl;
* }
* };
* int main()
* {
* Diffie\_Hellman obj;
* obj.getting\_a();
* obj.getPrivateKeys();
* obj.calculatePublicKeys();
* obj.generated\_SecretKey();
* obj.print();
* return 0;
* }

**Output:**

****

