Information and Network Security
2CSDE54
Practical 7

21BCE020

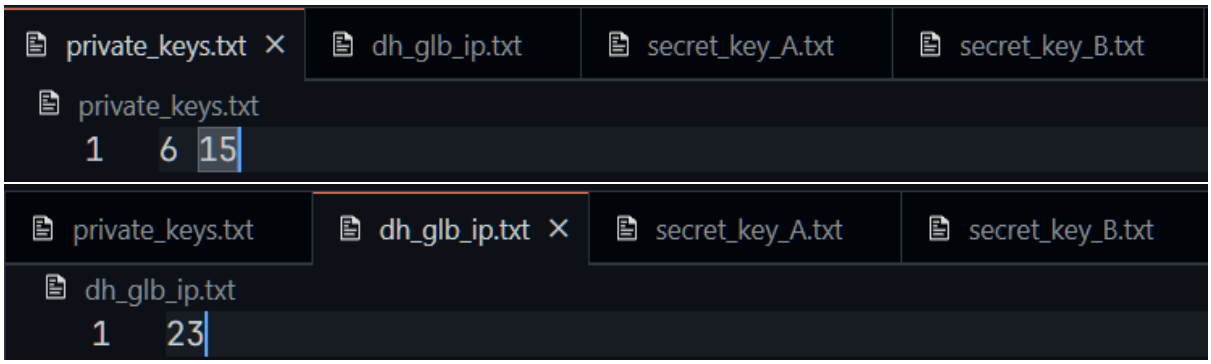
Implement the Diffie-Hellman Key Exchange algorithm.

```
#include <iostream>
#include <algorithm>
#include <string>
#include <fstream>
#include <math.h>
using namespace std;

int compute_power_mod(int b, int e, int m) {
    int res = 1; // identity element for multiplication
    b = b % m; // avoid overflow
    while (e > 0) {
        if (e % 2 == 1) // odd exponent
            res = (res * b) % m;
        e = e >> 1;
        b = (b * b) % m;
    }
    return res;
}


int primitive_root(int q) {
    for (int i = 2; i < q; ++i) {
        bool is_primitive = true; // track if i is primitive
root
        int res = 1;
        for (int j = 1; j < q - 1; ++j) {
            res = (res * i) % q;
            if (res == 1) {
                is_primitive = false; // not primitive root
                break;
            }
        }
        if (is_primitive) {
            return i;
```

```
        }
    }
    return -1;
}


int compute_Y(int X, int q) {
    int a = primitive_root(q);
    return compute_power_mod(a, X, q); //Y = a^X mod q
}

int gen_secret_key(int X, int Y, int q) {
    return compute_power_mod(Y, X, q); // K = Y^X mod q
}

int main() {
    ifstream f1("dh_glb_ip.txt");
    ifstream f2("private_keys.txt");
    ofstream f3("secret_key_A.txt");
    ofstream f4("secret_key_B.txt");
    int q; // a = 5
    int Xa, Xb;
    f1>>q;
    f2>>Xa>>Xb;
    int Ya = compute_Y(Xa, q);
    int Yb = compute_Y(Xb, q);
    int Ka = gen_secret_key(Xa, Yb, q);
    int Kb = gen_secret_key(Xb, Ya, q);
    f3<<"User A Public Key: "<<Ya<<endl;
    f4<<"User B Public Key: "<<Yb<<endl;
    cout<<"User A Public Key: "<<Ya<<endl;
    cout<<"User B Public Key: "<<Yb<<endl;
    cout<<"Secret Keys Generated: "<<Ka<<" and "<<Kb<<endl;
}
```

---

sText File Input:



Text File Output: