Program: 9 **Diffie-Hellman Key Exchange Algorithm**

Date:

AIM

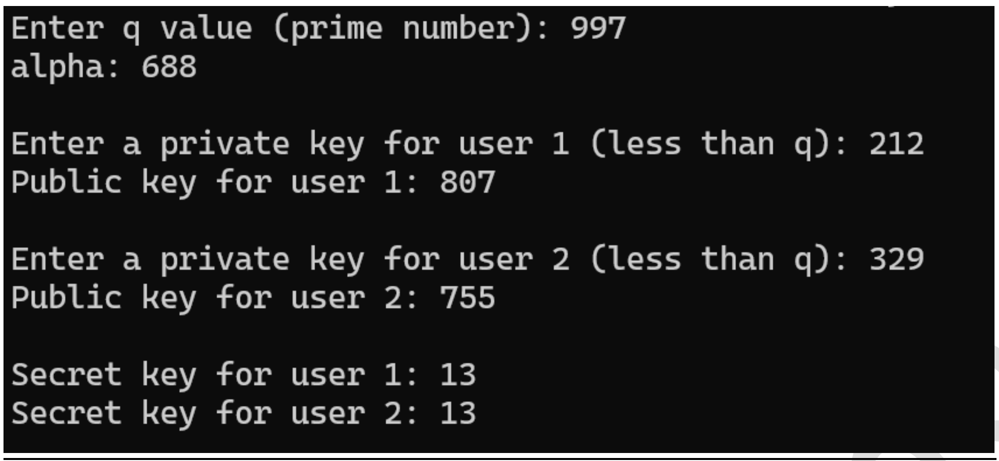
ALGORITHM

CODE

```
#include <iostream>
#include <vector>
#include <random>
using namespace std;

long int power(int a, int b, int mod) {
    long long int t;
    if(b==1)
        return a;
    t = power(a,b/2,mod);
    if(b%2==0)
        return (t*t)%mod;
    else
        return (((t*t)%mod)*a)%mod;
}

int isPrime(int n) {
    int is_prime = 1;
    int i;
    // 0 and 1 are not prime numbers
    if (n == 0 || n == 1) {
        is_prime = 1;
    }
    // loop to check if n is prime
    for (i = 2; i <= n/2; ++i) {
        if (n % i == 0) {
            is_prime = 0;
            break;
        }
    }
    return is_prime;
}
```

```
int calculateAlpha(int q) {
    int curAlpha = 1;
    vector<int> alpha_values;
    while (true) {
        int validAlpha = 1;
        vector<int> vec(q, 0);
        for (int i=1; i<q; i++) {
            int res = power(curAlpha, i, q);
            if (vec[res] == 1) {
                validAlpha = 0;
                break;
            }
            vec[res] = 1;
        }
        if (validAlpha) {
            alpha_values.push_back(curAlpha);
        }
        curAlpha++;
        if (curAlpha >= q) break;
    }
    std::random_device rd; // obtain a random number from hardware
    std::mt19937 gen(rd()); // seed the generator
    std::uniform_int_distribution<> distr(0, alpha_values.size()-1); // define the
range
    return alpha_values[distr(gen)];
}

int main() {
    int q;
    cout << "Enter q value (prime number): ";
    cin >> q;
    if (!isPrime(q)) {
        cout << "q must be a prime number!" << endl;
        return 1;
    }
    int alpha = calculateAlpha(q);
    cout << "alpha: " << alpha << endl;
    int xA, yA, xB, yB, k1, k2;
    cout << "\nEnter a private key for user 1 (less than q): ";
    cin >> xA;
    if (xA >= q) {
        cout << "private key must be less than q!" << endl;
        return 1;
    }
    yA = power(alpha, xA, q);
    cout << "Public key for user 1: " << yA << endl;
    cout << "\nEnter a private key for user 2 (less than q): ";
    cin >> xB;
    if (xB >= q) {
        cout << "private key must be less than q!" << endl;
        return 1;
    }
    yB = power(alpha, xB, q);
    cout << "Public key for user 2: " << yB << endl;
    k1 = power(yB, xA, q);
    k2 = power(yA, xB, q);
    cout << "\nSecret key for user 1: " << k1 << endl;
    cout << "Secret key for user 2: " << k2 << endl;
    return 0;
}
```

OUTPUT



```
Enter q value (prime number): 997
alpha: 688

Enter a private key for user 1 (less than q): 212
Public key for user 1: 807

Enter a private key for user 2 (less than q): 329
Public key for user 2: 755

Secret key for user 1: 13
Secret key for user 2: 13
```

RESULT

Thus, the program to implement secret key generation using diffie-hellman key exchange algorithm.