NIS

LAB - 4

ROll NO.: CE146

Name: I9CEU05159

ID NO.: Shubham Shingala

* AIM: write a Program to implement
  1) square and multiply function
  2) RSA Algorithm

1) square and multiply function

→ source code:

```
#include <iostream>
#define ll long long
using namespace std;

ll squareMultiply (ll base, ll exp, ll mod)
{
    // (base^ exp) % mod
    ll z=1;
    while(exp>0)
    {
        if (exp % 2 == 1)
            z=(z* base) % mod;
        exp = exp/2;
        base = (base * base) % mod;
    }
    return z;
}
```

```
int main () {
    ll num, exp, modeNum;
    cout << " Enter number, Exponential
     Number and mode Number : " ;
    cin >> num >> exp >> modeNum ;
    cout << num << " ^ " << exp << "mode"
        << modeNum << " : " <<
        SquareMultiply (num, exp, modeNum);
}
```

→ Test case - 1 :

Input : 424 24 34

Output : 424 ^ 24 mode 34 : 18

→ Test case - 2 :

Input : 94839 843 9832

Output : 94839 ^ 843 mode 9832 :

8439

2) RSA Algorithm :

→ Source code :

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;

ll squareMultiply ( ---) {
    // same as previous program function }

int randomNumberBetweenRange (int n, int m)
{
    int num;
    srand (time(0));
    while (true) {
        num = rand() % m;
        if (num > n)
            break;
    }
    return num;
}  // here n and m are not included.

bool millerRabin (int n) {
    if (n <= 1 || n == 4) return false;
    if (n <= 3) return true;
    int d = n-1, i=0, k, m, a;
    while (d % 2 == 0) {
        d /= 2; i++;
    }
```

```
K = i;
d = n-1;
m = d / pow(2, K);
a = RandomNumberBetweenRange(1, d);
int b = squaremultiply(a, m, n);
if (b % n == 1)
    return true; // n is prime
for (i=0; i<K; i++)
    if (b % n == d) //b % n == -1
        return true; // n is prime
    else
        b = (b*b) % n;
return false; // n is composite
}

int multiplicative_inverse(int a, int b){
int q, r, t, t1=0, t2=1, r1, r2;
r1 = b;
r2 = a;
while(r2 > 0){
    q = r1/r2;
    r = r1 - q*r2;
    r1 = r2; r2 = r;
    t = t1 - q*t2;
    t1 = t2; t2 = t;
}
if (r1 == 1){
    if(t1 < 0)
        t1 += b;
```

```
        return t1;
    }
    return 0;
}

void keyGenerate (ll * key)
{
    int p, q, n;
    srand(time(0));
    while (true) {
        p = rand(); q = rand();
        if ( p != q && millerRabin(p)
                && millerRobin (q))
            break;
    }
    n = p * q;
    int fi = (p-1) * (q-1), e,d;
    while(true) {
        e = random Number Between Range
        // e = 5;                    (1, fi);
        d = multiplicative_inverse(e, fi);
        if (d != 0)
            break;
    }
    key [0] = e; key [1] = d; key [2] = n;
    key [3] = p; key [4] = q;
}
```

```
ll encrypt (ll num, ll publicKey, ll modeNum)
{
    return squareMultiply (num, publicKey, modeNum);
}

ll decrypt (ll encryptNum, ll privateKey, ll modeNum)
{
    return squareMultiply (encryptNum,
                privateKey, modeNum);
}

int main ( ) {
    ll num;
    cout << " Enter Number : ";
    cin >> num;
    ll key [5];
    keyGenerate (key) ; //key(0) = public key
    // key(1) = private key , key2 = n key3=p, key4=q;
    cout << " public key = " << key [0] <<
    " , private key = " << key [1] <<
    " , n = " << key [2] << endl;
    cout << " p = " << key [3] << " q = " <<
        key [4] << endl;
    ll encryptNum = encrypt (num, key [0], key[2]);
    cout << " encrypt Number : " << encryptNum
    << endl << " decrypt Number :"
    << decrypt (encryptNum, key [1], key [2]);
}
```

→ Test case - 1 :

Input : Number = 32

Output : Public Key = 27793
Private Key = 73522097
n = 433793557
p = 26417
q = 16421
encrypt Number = 106573227
decrypt Number = 32

→ Test case - 2 :

Input : Number = 23

Output : Public Key = 13473
Private Key = 61155937
n = 71612509
p = 3041
q = 23549
encrypt Number = 27907143
decrypt Number = 23