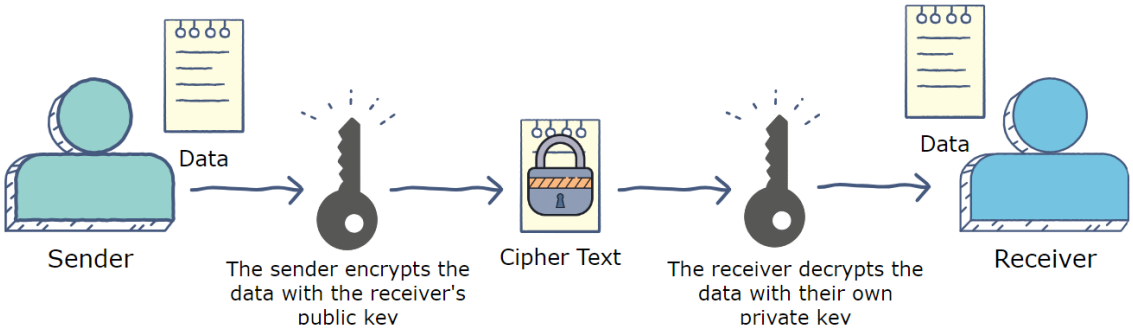
# Description:

**Implementation of RSA Algorithm.**

The RSA algorithm is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.

The following illustration highlights how asymmetric cryptography works:




## How it works

The RSA algorithm ensures that the keys, in the above illustration, are as secure as possible. The following steps highlight how it works:

### 1. Generating the keys

1. Select two large prime numbers, $p$ and $q$. The prime numbers need to be large so that they will be difficult for someone to figure out.
2. Calculate $n = p \times q$.
3. Calculate the Eulers *totient* function; $\phi(n)=(p-1)(q-1)$.
4. Select an integer $e$, such that $e$ is *co-prime* to $\phi(n)$ and $1<e<\phi(n)$. The pair of numbers $(n,e)$ makes up the public key.
5. Calculate $d$ such that $e \cdot d \equiv 1 \bmod \phi(n)$ i.e. $(e \cdot d) \bmod \phi(n) = 1$

   $d$ can be found using the *extended euclidean algorithm*. The pair $(n,d)$ makes up the private key. i.e. we need to find multiplicative inverse of $e \bmod \phi(n)$

### 2. Encryption

Given a plaintext $M$, represented as a number, the ciphertext $C$ is calculated as: $C = M^e \bmod n$.

### 3. Decryption

Using the private key $(n,d)$, the plaintext can be found using: $P = C^d \bmod n$.

**Example:**

Choose p = 3 and q = 11
Compute n = p * q = 3 * 11 = 33
Compute φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20

Choose e such that $1 < e < \varphi(n)$ and e and $\varphi$ (n) are coprime. Let $e = 7$
Compute a value for d such that $(d * e) \% \varphi(n) = 1$. One solution is $d = 3$ $[(3 * 7) \% 20 = 1]$
Public key is (e, n) => (7, 33)
Private key is (d, n) => (3, 33)
The encryption of $m = 2$ is $c = 27 \% 33 = 29$
The decryption of $c = 29$ is $m = 293 \% 33 = 2$

How to find multiplicative inverse $e \bmod \phi(n)$

**In the below example we have calculated for 3 mod 5**

| Q | A | B | R | T1 | T2 | T |
|---|---|---|---|---|---|---|
| 1 | 5 | 3 | 2 | 0 | 1 | -1 |
| 1 | 3 | 2 | 1 | 1 | -1 | 2 |
| 2 | 2 | 1 | 0 | -1 | 2 | -5 |
| X | 1 | 0 | X | 2 | -5 | X |
| | | | | | | |
| | | | | | | |

Note: A>B
Initial value:
T1=0
T2=1
T=T1-(T2XQ)

**METHODOLOGY FOLLOWED:**

**File1: Ras.cpp**

```
#include <iostream>
#include <bits/stdc++.h>
#include <fstream>

using namespace std;
```

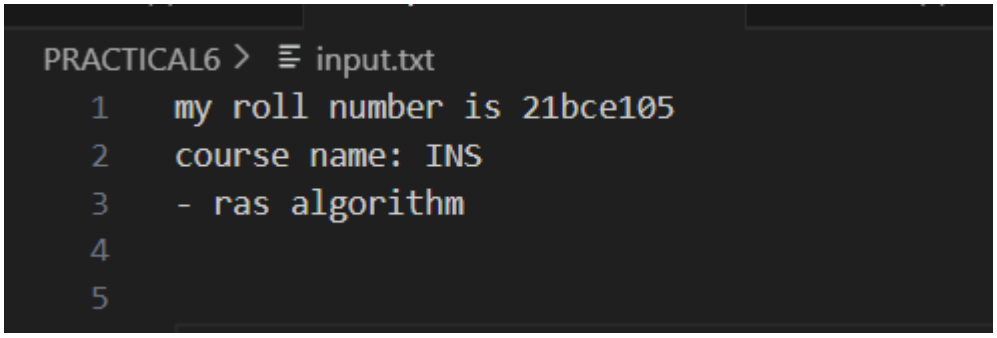
```cpp
const int N = 10e3 + 1;
bool prime[N + 1];

vector<int> prime_numbers;

// power function calculate  ( x to the power y )%mod
long long int power(long long int x, long long int y, long long int mod)
{
    if (y == 0)
    {
        return 1 % mod;
    }
    if (y == 1)
    {
        return x % mod;
    }

    if (y % 2 == 0)
    {
        long long int a = power(x, y / 2, mod);
        return (a * a) % mod;
    }
    else
    {
        return (power(x, y / 2, mod) * power(x, y - y / 2, mod)) % mod;
    }
}

long long int GCD(long long int a, long long int b)
{
    if (a == 0)
        return b;
    if (b == 0)
        return a;

    // base case
    if (a == b)
        return a;

    // a is greater
    if (a > b)
        return GCD(a - b, b);

    return GCD(a, b - a);
}

int main()
```

```
{
    for (int i = 0; i <= N; i++)
    {
        prime[i] = true;
    }
    prime[0] = false;
    prime[1] = false;
    prime[2] = true;

    for (int i = 2; i * i <= N; i++)
    {
        if (prime[i])
        {
            for (int j = 2 * i; j <= N; j += i)
            {
                prime[j] = false;
            }
        }

        for (int i = 0; i <= N; i++)
        {
            if (prime[i])
            {
                prime_numbers.push_back(i);
            }
        }
    }

    long long int p = prime_numbers[20];
    long long int q = prime_numbers[25];

    long long int n = p * q;
    long long int Qn = (p - 1) * (q - 1);
    long long int e = -1;

    for (long long int i = 2; i < Qn; i++)
    {
        if (GCD(i, Qn) == 1)
        {
            e = i;
            break;
        }
    }

    long long int Q, A, B, R, T1, T2, T;

    A = Qn;
    B = e;
```

```
    long long int a = A;
    long long int b = B;
    Q = A / B;
    R = A % B;

    cout << A << "A " << B << "B\n";

    T1 = 0; // initially take t1 =0
    T2 = 1; // initially take t2 =1

    T = T1 - T2 * Q;

    // cout << Q << " " << A << " " << B << " " << R << " " << T1 << " " << T2
<< " " << T << "***++\n";

    while ((R != 0))
    {
        A = B;
        B = R;
        Q = A / B;
        R = A % B;
        T1 = T2; // initially take t1 =0
        T2 = T;  // initially take t2 =1
        T = T1 - T2 * Q;
        // cout << Q << " " << A << " " << B << " " << R << " " << T1 << " "
<< T2 << " " << T << "***++\n";
    }

    A = B;
    B = R;
    T1 = T2; // initially take t1 =0
    T2 = T;  // initially take t2 =1
    T = T1 - T2 * Q;

    long long int d = T1;

    if (d < 0)
    {
        d = a + d; // positive multiplicative inverse
    }

    ifstream fin;
    fin.open("input.txt");

    ofstream fout;
    fout.open("output.txt");

    char ch;
```

```
    while (fin.get(ch))
    {
        long long int m = ch;
        cout << ch << " " << m << " " << e << " pow: " << power(m, e, n) <<
"\n";
        long long int cipher = power(m, e, n); // power function calculate
(m^e)%n
        cout << "\n"
             << cipher << "\n";
        fout << cipher << " ";
    }

    fout.close();
    fin.close();

    ifstream fin2;
    fin2.open("output.txt");

    ofstream fout2;
    fout2.open("doutput.txt");

    long long int c;
    while (fin2 >> c)
    {

        cout << c << "\n";
        cout << c << " " << d << " pow: " << power(c, d, n) << "\n";
        long long int decrypted = power(c, d, n); // power function calculate
(c^d)%n
        fout2 << (char)decrypted;
    }
    fout2.close();
    fin2.close();

    return 0;
}
```

File2: input.txt ( plain text)




File2: output.txt (cipher text) we store encrypted text in this file. Here I not store characters because it creat an issue. (Reason is that character -> 0 to 255 ASCII value in c++)

700 484 1403 1116 117 930 930 1403 1124 1998 700 242 1529 1116 1403 1302 718 1403 1631 553 242 75 1529 553 1860 416 1000 75 117 1998 1116 718 1529 1403 1124 997 700 1529 649 1403 91 1986 944 1000 1212 1403 1116 997 718 1403 997 930 1225 117 1116 1302 1010 1997 700 1000 1000 1000

File3: doutput.txt