Roll no. 210701313

**Ex. No.: 4** **Date:**

**RSA**

**Aim:**

To implement RSA asymmetric key cryptosystem using C.

**Algorithm:**

1. Select two large prime numbers p and q
2. Compute n=pxq
3. Choose system modulus: Ø(n)=(p-1)x(q-1)
4. Select a random encryption key e such that gcd(e,Ø(n)=1
5. Decrypt by computing d=1 mod Ø(n)
6. Print the public key{e,n}
7. Print the private key{d,n}

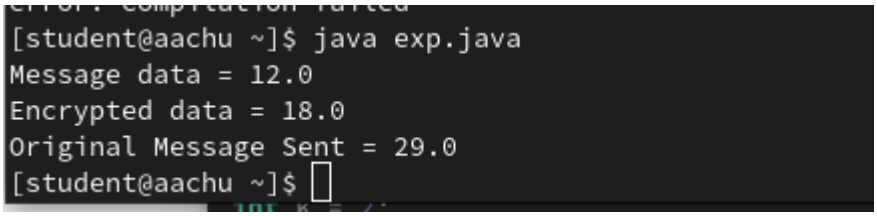
**Program Code:**

```
#include <stdio.h>
#include <math.h>
int power(int,unsigned int,int);
int gcd(int,int);
int multiplicativeInverse(int,int,int);
int main()
{
  int p,q,n,e,d,phi,M,C;

  printf("\nEnter two prime numbers p and q that are not equal : ");
  scanf("%d %d",&p,&q);
  n = p * q;
  phi = (p - 1)*(q - 1);
  printf("Phi(%d) = %d",n,phi);
  printf("\nEnter the integer e : ");
  scanf("%d",&e);
  if(e >= 1 && e < phi)
  {
        if(gcd(phi,e)!=1)
        {
              printf("\nChoose proper value for e !!!\n");
              return 1;
        }
  }

  //Key Generation
  d = multiplicativeInverse(e,phi,n);

  printf("\nPublic Key PU = {%d,%d}",e,n);
```

```
  printf("\nPrivate Key PR = {%d,%d}",d,n);

  //Encryption
  printf("\nMessage M = ");
  scanf("%d",&M);
  C = power(M,e,n);
  printf("\nCiphertext C = %d \n",C);

  //Decryption
  M = power(C,d,n);
  printf("\nDecrypted Message M = %d \n",M);


  return 0;
}


int power(int x, unsigned int y, int p)
{
    int res = 1;      // Initialize result

    x = x % p; // Update x if it is more than or equal to p

    while (y > 0)
    {
        // If y is odd, multiply x with
        result if (y & 1)
            res = (res*x) % p;


        // y must be even now
        y = y>>1;  // y = y/2 x
        = (x*x) % p;
    }
    return res;
}


int gcd ( int a, int b )
{
  int c;
  while ( a != 0 )
  {
          c = a;
          a = b % a;
          b = c;
  }
  return b;
}
```

```
int multiplicativeInverse(int a, int b, int n)
{
        int sum,x,y;

    for(y=0;y<n;y++)
    {
        for(x=0;x<n;x++)
        {
            sum=a*x + b*(-y);
            if(sum==1)
                return x;
        }
    }
}
```

**Output:**




**Result:**