Caeser Cipher code :

```
#include <stdio.h>

#include <ctype.h>

#include<string.h>


char enc(char ch, int key)

{

  if (isalpha(ch))

  {

    char b = islower(ch) ? 'a' : 'A';  // Added spaces around operators for readability

    ch = (ch - b + key) % 26 + b;     // Added spaces around operators for readability

  }

  return ch;

}


int main()  // Changed 'void main()' to 'int main()'

{

  char pt[100];

  int key, i;  // Added a space after the comma for readability


  printf("Enter the plain text: ");

  scanf("%s",pt);


  printf("Enter the key value: ");

  scanf("%d", &key);


  for (i = 0; pt[i] != '\0'; i++)

    pt[i] = enc(pt[i], key);
```

```
    printf("Encrypted text: %s\n", pt);  // Replaced non-ASCII space with a regular space

    return 0;  // Added return 0 to the main function
}
```

Diffie Hellman Code :

```
#include <stdio.h>
#include <math.h>

long long powerMod(long long base, long long exp, long long mod) {
    long long result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1)
            result = (result * base) % mod;
        exp = exp >> 1;
        base = (base * base) % mod;
    }
    return result;
}

int main() {
    long long p, g, a, b;
    long long A, B, sharedKeyA, sharedKeyB;

    printf("Enter a large prime number (p): ");
    scanf("%lld", &p);
```

```
    printf("Enter a primitive root modulo p (g): ");
    scanf("%lld", &g);
    printf("Enter the private key of the first party (a): ");
    scanf("%lld", &a);
    printf("Enter the private key of the second party (b): ");
    scanf("%lld", &b);

    A = powerMod(g, a, p);
    B = powerMod(g, b, p);

    sharedKeyA = powerMod(B, a, p);
    sharedKeyB = powerMod(A, b, p);

    printf("Public Key of First Party (A): %lld\n", A);
    printf("Public Key of Second Party (B): %lld\n", B);
    printf("Shared Secret Key computed by First Party: %lld\n", sharedKeyA);
    printf("Shared Secret Key computed by Second Party: %lld\n", sharedKeyB);

    return 0;
}
```

RSA_CODE :

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>

long int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;
char msg[100];
```

```
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
//void decrypt();

int main()
{
   printf("ENTER FIRST PRIME NUMBER: ");
   scanf("%ld", &p);
   flag = prime(p);
   if (flag == 0 || p == 1)
   {
      printf("WRONG INPUT\n");
      exit(1);
   }

   printf("ENTER ANOTHER PRIME NUMBER: ");
   scanf("%ld", &q);
   flag = prime(q);
   if (flag == 0 || q == 1 || p == q)
   {
      printf("WRONG INPUT\n");
      exit(1);
   }

   printf("ENTER MESSAGE: ");
```

```
    scanf(" %[^\n]s", msg);
    for (i = 0; i < strlen(msg); i++)
        m[i] = msg[i];

    n = p * q;
    t = (p - 1) * (q - 1);

    ce();

    printf("\nPOSSIBLE VALUES OF e AND d ARE:\n");
    for (i = 0; i < j - 1; i++)
        printf("%ld\t%ld\n", e[i], d[i]);

    encrypt();
    //decrypt();

    return 0;
}

int prime(long int pr)
{
    int i;
    if (pr == 1)
        return 0;

    for (i = 2; i <= sqrt(pr); i++)
    {
        if (pr % i == 0)
```

```
        return 0;
    }
    return 1;
}

void ce()
{
    int k = 0;
    for (i = 2; i < t; i++)
    {
        if (t % i == 0)
            continue;
        flag = prime(i);
        if (flag == 1 && i != p && i != q)
        {
            e[k] = i;
            flag = cd(e[k]);
            if (flag > 0)
            {
                d[k] = flag;
                k++;
            }
            if (k == 99)
                break;
        }
    }
}
```

```
long int cd(long int x)
{
   long int k = 1;
   while (1)
   {
      k = k + t;
      if (k % x == 0)
         return (k / x);
   }
}

void encrypt()
{
   long int pt, ct, key = e[0], k, len;
   i = 0;
   len = strlen(msg);
   while (i < len)
   {
      pt = m[i];
      pt = pt - 96;
      k = 1;
      for (j = 0; j < key; j++)
      {
         k = k * pt;
         k = k % n;
      }
      temp[i] = k;
      ct = k + 96;
```

```
    en[i] = ct;
    i++;
  }
  en[i] = -1;
  printf("\nTHE ENCRYPTED MESSAGE IS:\n");
  for (i = 0; en[i] != -1; i++)
    printf("%c", (char)en[i]);
}

/*
void decrypt()
{
  long int pt, ct, key = d[0], k;
    i = 0;
  while (en[i] != -1)
  {
    ct = temp[i];
    k = 1;
    for (j = 0; j < key; j++)
    {
      k = k * ct;
      k = k % n;
    }
    pt = k + 96;
    m[i] = pt;
    i++;
  }
  m[i] = -1;
```

```
   printf("\nTHE DECRYPTED MESSAGE IS:\n");
   for (i = 0; m[i] != -1; i++)
      printf("%c", (char)m[i]);
}
*/
```

Transpositional Cipher Code :

```
#include<stdio.h>

int check(int x, int y) {
   return (x % y == 0) ? 0 : (y * ((x / y) + 1)) - x;
}

int main() {
   int l1, i, j, d, l2;

   printf("\nEnter the length of the key: ");
   scanf("%d", &l1);
   int sequence[l1], order[l1];

   printf("\nEnter the sequence key: ");
   for (i = 0; i < l1; ++i) scanf("%d", &sequence[i]);

   for (i = 1; i <= l1; ++i)
      for (j = 0; j < l1; ++j)
         if (sequence[j] == i) order[i-1] = j;

   printf("\nEnter the depth: ");
```

```
  scanf("%d", &d);

  printf("\nEnter the length of String without spaces: ");
  scanf("%d", &l2);
  int temp1 = check(l2, l1);
  int r = (l2 + temp1) / l1;

  char p[l2 + temp1], p1[r][l1];

  if (temp1 > 0)
    printf("\nYou need to enter %d bogus characters. Enter total %d characters: ",
temp1, l2 + temp1);
  else
    printf("\nEnter the string: ");

  for (i = 0; i < l2 + temp1; ++i) scanf(" %c", &p[i]);

  while (d-- > 0) {
    int count = 0;
    for (i = 0; i < r; ++i)
      for (j = 0; j < l1; ++j)
        p1[i][j] = p[count++];

    for (i = 0; i < r; ++i) {
      for (j = 0; j < l1; ++j) printf("%c ", p1[i][j]);
      printf("\n");
    }
```

```
      count = 0;
      for (i = 0; i < l1; ++i)
         for (j = 0; j < r; ++j)
            p[count++] = p1[j][order[i]];

      for (i = 0; i < l2 + temp1; ++i) printf("%c ", p[i]);
      printf("\n");
   }
}
```

```
/* OUTPUT
[prat@localhost Desktop]$ gcc transposition.c
[prat@localhost Desktop]$ ./a.out

Enter the length of the key. 7

Enter the sequence key. 4 3 1 2 5 6 7

Enter the depth. 2

Enter the length of String without spaces . 23

You need to enter 5 bogus characters.So enter total 28 characters.
attackpostponeduntiltwoamxyz
```

a t t a c k p

o s t p o n e

d u n t i l t

w o a m x y z

t t n a a p t m t s u o a o d w c o i x k n l y p e t z

t t n a a p t

m t s u o a o

d w c o i x k

n l y p e t z

n s c y a u o p t t w l t m d n a o i e p a x t t o k z

*/