NSC LAB

Week-1: Caeser Cipher

```
#include <iostream>
using namespace std;
string encrypt(string text, int s)
{
    string result = "";
    for (int i = 0; i < text.length(); i++)
    {
        if (isupper(text[i]))
            result += char(int(text[i] + s - 65) % 26 + 65);
        else
            result += char(int(text[i] + s - 97) % 26 + 97);
    }
    return result;
}
string decrypt(string text, int s)
{
    string result = "";
    for (int i = 0; i < text.length(); i++)
    {
        if (isupper(text[i]))
        {
            int res=int(text[i] - s - 65) % 26 + 65;
            if(res<65)
                result += char(res+26);
            else
                result += char(res);
        }
        else
        {
            int res=int(text[i] - s - 97) % 26 + 97;
            if(res<97)
                result += char(res+26);
            else
                result += char(res);
        }
    }
    return result;
}
int main()
{
    string text;
    int key;
    cout<<"Enter a string to encrypt:";
    cin>>text;
    cout<<"Enter the key:";
    cin>>key;
    cout << "Text : " << text;
    cout << "\nKey: " << key;
    string result = encrypt(text,key);
    cout << "\nEncrypted text is: " << encrypt(text,key);
    cout << "\nDecrypted text is: " << decrypt(result,key);
    return 0;
}
```

Output:

```
Enter a string to encrypt:NETWORKSECURITY

Enter the key:4

Text : NETWORKSECURITY

Key: 4

Encrypted text is: RIXASVOWIGYVMXC

Decrypted text is: NETWORKSECURITY
```

WEEK-2: Hill Cipher

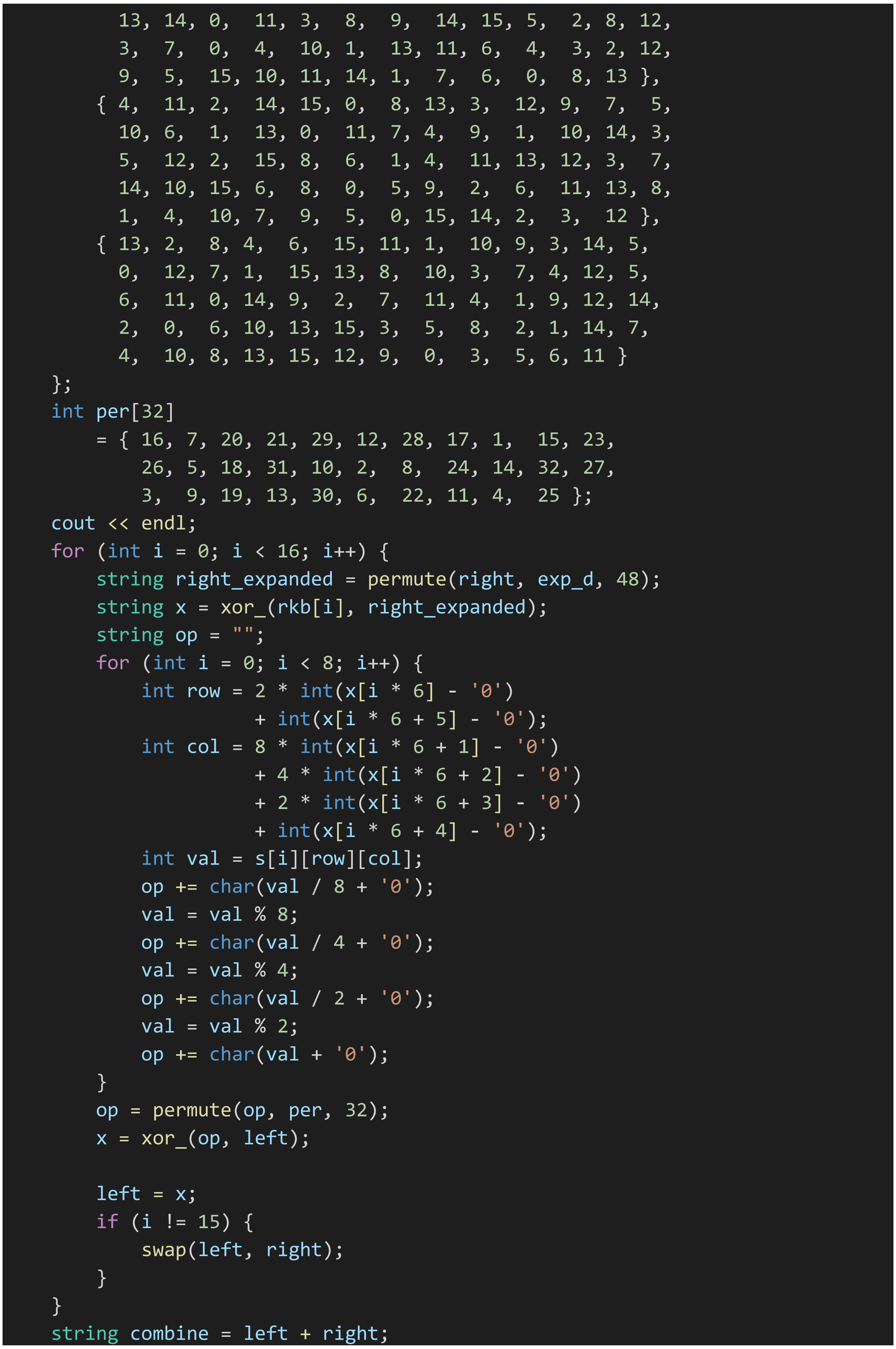
```
#include<bits/stdc++.h>
using namespace std;
int b[3][3],de[3][3],messageVector[1][3],keyMatrix[3][3],inverseMatrix[3][3];
string text,key,res;
int calcInverse(int det)
{
    int i=1;
    while((det*i)%26!=1)
    {
        i++;
    }
    return i;
}
void getkeyMatrix(string key,int keyMatrix[][3])
{
    int k = 0;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            keyMatrix[i][j] = (key[k]) % 65;
            k++;
        }
    }
}
void encrypt(int cipherkeyMatrix[][3],int keyMatrix[][3],int 
messageVector[][3])
{
    int x, i, j;
    for (j = 0; j < 3; j++)
    {
        for (i = 0; i < 1; i++)
        {
            cipherkeyMatrix[i][j] = 0;
            for (x = 0; x < 3; x++)
            {
                cipherkeyMatrix[i][j] +=
                    keyMatrix[x][j] * messageVector[0][x];
            }
            cipherkeyMatrix[i][j] = cipherkeyMatrix[i][j] % 26;
        }
    }
}
void HillCipher(string message, string key)
{
    int keyMatrix[3][3];
    getkeyMatrix(key, keyMatrix);
    int i=0;
    while(i<message.length())
    {
        for(int j=0;j<3;j++)
        {
        messageVector[0][j] = (message[i]) % 65;
        i++;
        }
    int cipherkeyMatrix[1][3];
    encrypt(cipherkeyMatrix, keyMatrix, messageVector);
    for (int i = 0; i < 3; i++)
    {
        res += cipherkeyMatrix[0][i] + 65;
    }
    res+="";
    }
    cout << " Ciphertext:" << res;
}
void inversekeyMatrix(string key) {
   getkeyMatrix(key, keyMatrix);
   int det=0;
   for(int i = 0; i < 3; i++)
    det = det + (keyMatrix[0][i] * (keyMatrix[1][(i+1)%3] *
keyMatrix[2][(i+2)%3] - keyMatrix[1][(i+2)%3] * keyMatrix[2][(i+1)%3]));
    cout<<det<<endl;
    if(det<0)
    {
        det=det%26+26;
        cout<<det;
    }
    cout<<"\n\nInverse of keyMatrix is: \n";
    for(int i = 0; i < 3; i++){
    for(int j = 0; j < 3; j++)
    {
    inverseMatrix[i][j]=((keyMatrix[(j+1)%3][(i+1)%3] *
keyMatrix[(j+2)%3][(i+2)%3]) - (keyMatrix[(j+1)%3][(i+2)%3] *
keyMatrix[(j+2)%3][(i+1)%3]))%26;
    if(inverseMatrix[i][j]<0)
    inverseMatrix[i][j]+=26;
    }
    }
    int x=calcInverse(det);
   for(int i = 0; i < 3; i++){
    for(int j = 0; j < 3; j++)
    {
        inverseMatrix[i][j]*=x;
        inverseMatrix[i][j]%=26;
    }
    }
}
void decrypt(string key)
{
   inversekeyMatrix(key);
   int x, i, j;
   string res1;
   int a[res.length()],c[1][3],cipherkeyMatrix1[1][3];
   for(int i=0;i<res.length();i++)
   {
       a[i]=res[i]%65;
   }
   for(int b=0;b<res.length();b+=3)
   {
       c[0][0]=a[b];
       c[0][1]=a[b+1];
       c[0][2]=a[b+2];
       for (j = 0; j < 3; j++)
       {
            for (i = 0; i < 1; i++)
            {
                cipherkeyMatrix1[i][j] = 0;
                for (x = 0; x < 3; x++)
                {
                    cipherkeyMatrix1[i][j] +=inverseMatrix[x][j] * c[0][x];
                }
                cipherkeyMatrix1[i][j] = cipherkeyMatrix1[i][j] % 26;
            }
        }
       for (int i = 0; i < 3; i++)
      {
            res1 += cipherkeyMatrix1[0][i] + 65;
        }
        res1+=" ";
   }
   cout<<res1<<" ";
}
int main()
{
    cout<<"Enter a string to encrypt:";
    cin>>text;
    if(text.length()%3!=0)
    {
        int s=text.length()%3;
        for(int i=0;i<3-s;i++)
        text+="X";
    }
    cout<<"Enter the key:";
    cin>>key;
    // string key = "RRFVSVCCT";
    HillCipher(text, key);
   decrypt(key);
    return 0;
}
```

Output:

```
Enter a string to encrypt:PAYMOREMONEY

Enter the key:RRFVSVCCT

Cipher text is:RRLMWBKASPDH

Decrypted text is:PAY MOR EMO NEY
```

WEEK-3: DES

```cpp
#include <bits/stdc++.h>
using namespace std;
string hex2bin(string s)
{
    unordered_map<char, string> mp;
    mp['0'] = "0000";
    mp['1'] = "0001";
    mp['2'] = "0010";
    mp['3'] = "0011";
    mp['4'] = "0100";
    mp['5'] = "0101";
    mp['6'] = "0110";
    mp['7'] = "0111";
    mp['8'] = "1000";
    mp['9'] = "1001";
    mp['A'] = "1010";
    mp['B'] = "1011";
    mp['C'] = "1100";
    mp['D'] = "1101";
    mp['E'] = "1110";
    mp['F'] = "1111";
    string bin = "";
    for (int i = 0; i < s.size(); i++) {
        bin += mp[s[i]];
    }
    return bin;
}
string bin2hex(string s)
{
    unordered_map<string, string> mp;
    mp["0000"] = "0";
    mp["0001"] = "1";
    mp["0010"] = "2";
    mp["0011"] = "3";
    mp["0100"] = "4";
    mp["0101"] = "5";
    mp["0110"] = "6";
    mp["0111"] = "7";
    mp["1000"] = "8";
    mp["1001"] = "9";
    mp["1010"] = "A";
    mp["1011"] = "B";
    mp["1100"] = "C";
    mp["1101"] = "D";
    mp["1110"] = "E";
    mp["1111"] = "F";
    string hex = "";
    for (int i = 0; i < s.length(); i += 4) {
        string ch = "";
        ch += s[i];
        ch += s[i + 1];
        ch += s[i + 2];
        ch += s[i + 3];
        hex += mp[ch];
    }
    return hex;
}

string permute(string k, int* arr, int n)
{
    string per = "";
    for (int i = 0; i < n; i++) {
        per += k[arr[i] - 1];
    }
    return per;
}
string shift_left(string k, int shifts)
{
    string s = "";
    for (int i = 0; i < shifts; i++) {
        for (int j = 1; j < 28; j++) {
            s += k[j];
        }
        s += k[0];
        k = s;
        s = "";
    }
    return k;
}

string xor_(string a, string b)
{
    string ans = "";
    for (int i = 0; i < a.size(); i++) {
        if (a[i] == b[i]) {
            ans += "0";
        }
        else {
            ans += "1";
        }
    }
    return ans;
}
string encrypt(string pt, vector<string> rkb,
               vector<string> rk)
{
    pt = hex2bin(pt);
    int initial_perm[64]
        = { 58, 50, 42, 34, 26, 18, 10, 2,  60, 52, 44,
            36, 28, 20, 12, 4,  62, 54, 46, 38, 30, 22,
            14, 6,  64, 56, 48, 40, 32, 24, 16, 8,  57,
            49, 41, 33, 25, 17, 9,  1,  59, 51, 43, 35,
            27, 19, 11, 3,  61, 53, 45, 37, 29, 21, 13,
            5,  63, 55, 47, 39, 31, 23, 15, 7 };
    pt = permute(pt, initial_perm, 64);
    string left = pt.substr(0, 32);
    string right = pt.substr(32, 32);
    int exp_d[48]
        = { 32, 1,  2,  3,  4,  5,  4,  5,  6,  7,  8,  9,
            8,  9,  10, 11, 12, 13, 12, 13, 14, 15, 16, 17,
            16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25,
            24, 25, 26, 27, 28, 29, 28, 29, 30, 31, 32, 1 };
    int s[8][4][16] = {
        { 14, 4,  13, 1, 2,  15, 11, 8,  3,  10, 6,  12, 5,
          9,  0,  7,  0, 15, 7,  4,  14, 2,  13, 1,  10, 6,
          12, 11, 9,  5, 3,  8,  4,  1,  14, 8,  13, 6,  2,
          11, 15, 12, 9, 7,  3,  10, 5,  0,  15, 12, 8,  2,
          4,  9,  1,  7, 5,  11, 3,  14, 10, 0,  6,  13 },
        { 15, 1,  8,  14, 6,  11, 3, 4,  9,  7,  2,  13, 12,
          0,  5,  10, 3,  13, 4,  7, 15, 2,  8,  14, 12, 0,
          1,  10, 6,  9,  11, 5,  0, 14, 7,  11, 10, 4,  13,
          1,  5,  8,  12, 6,  9,  3, 2,  15, 13, 8,  10, 1,
          3,  15, 4,  2,  11, 6,  7, 12, 0,  5,  14, 9 },

        { 10, 0,  9,  14, 6,  3,  15, 5,  1,  13, 12,
          7,  11, 4,  2,  8,  13, 7,  0,  9,  3,  4,
          6,  10, 2,  8,  5,  14, 12, 11, 15, 1,  13,
          6,  4,  9,  8,  15, 3,  0,  11, 1,  2,  12,
          5,  10, 14, 7,  1,  10, 13, 0,  6,  9,  8,
          7,  4,  15, 14, 3,  11, 5,  2,  12 },
        { 7,  13, 14, 3,  0,  6,  9,  10, 1,  2, 8,  5,  11,
          12, 4,  15, 13, 8,  11, 5,  6,  15, 0, 3,  4,  7,
          2,  12, 1,  10, 14, 9,  10, 6,  9,  0, 12, 11, 7,
          13, 15, 1,  3,  14, 5,  2,  8,  4,  3, 15, 0,  6,
          10, 1,  13, 8,  9,  4,  5,  11, 12, 7, 2,  14 },
        { 2,  12, 4, 1,  7,  10, 11, 6, 8,  5,  3,  15, 13,
          0,  14, 9, 14, 11, 2,  12, 4, 7,  13, 1,  5,  0,
          15, 10, 3, 9,  8,  6,  4,  2, 1,  11, 10, 13, 7,
          8,  15, 9, 12, 5,  6,  3,  0, 14, 11, 8,  12, 7,
          1,  14, 2, 13, 6,  15, 0,  9, 10, 4,  5,  3 },
        { 12, 1,  10, 15, 9,  2,  6,  8,  0,  13, 3, 4, 14,
          7,  5,  11, 10, 15, 4,  2,  7,  12, 9,  5, 6, 1,
```



```
          13, 14, 0,  11, 3,  8,  9,  14, 15, 5,  2, 8, 12,
          3,  7,  0,  4,  10, 1,  13, 11, 6,  4,  3, 2, 12,
          9,  5,  15, 10, 11, 14, 1,  7,  6,  0,  8, 13 },
        { 4,  11, 2,  14, 15, 0,  8, 13, 3,  12, 9,  7,  5,
          10, 6,  1,  13, 0,  11, 7, 4,  9,  1,  10, 14, 3,
          5,  12, 2,  15, 8,  6,  1, 4,  11, 13, 12, 3,  7,
          14, 10, 15, 6,  8,  0,  5, 9,  2,  6,  11, 13, 8,
          1,  4,  10, 7,  9,  5,  0, 15, 14, 2,  3,  12 },
        { 13, 2,  8, 4,  6,  15, 11, 1,  10, 9, 3, 14, 5,
          0,  12, 7, 1,  15, 13, 8,  10, 3,  7, 4, 12, 5,
          6,  11, 0, 14, 9,  2,  7,  11, 4,  1, 9, 12, 14,
          2,  0,  6, 10, 13, 15, 3,  5,  8,  2, 1, 14, 7,
          4,  10, 8, 13, 15, 12, 9,  0,  3,  5, 6, 11 }
    };
    int per[32]
        = { 16, 7, 20, 21, 29, 12, 28, 17, 1,  15, 23,
            26, 5, 18, 31, 10, 2,  8,  24, 14, 32, 27,
            3,  9, 19, 13, 30, 6,  22, 11, 4,  25 };
    cout << endl;
    for (int i = 0; i < 16; i++) {
        string right_expanded = permute(right, exp_d, 48);
        string x = xor_(rkb[i], right_expanded);
        string op = "";
        for (int i = 0; i < 8; i++) {
            int row = 2 * int(x[i * 6] - '0')
                      + int(x[i * 6 + 5] - '0');
            int col = 8 * int(x[i * 6 + 1] - '0')
                      + 4 * int(x[i * 6 + 2] - '0')
                      + 2 * int(x[i * 6 + 3] - '0')
                      + int(x[i * 6 + 4] - '0');
            int val = s[i][row][col];
            op += char(val / 8 + '0');
            val = val % 8;
            op += char(val / 4 + '0');
            val = val % 4;
            op += char(val / 2 + '0');
            val = val % 2;
            op += char(val + '0');
        }
        op = permute(op, per, 32);
        x = xor_(op, left);

        left = x;
        if (i != 15) {
            swap(left, right);
        }
    }
    string combine = left + right;
    int final_perm[64]
        = { 40, 8,  48, 16, 56, 24, 64, 32, 39, 7,  47,
            15, 55, 23, 63, 31, 38, 6,  46, 14, 54, 22,
            62, 30, 37, 5,  45, 13, 53, 21, 61, 29, 36,
            4,  44, 12, 52, 20, 60, 28, 35, 3,  43, 11,
            51, 19, 59, 27, 34, 2,  42, 10, 50, 18, 58,
            26, 33, 1,  41, 9,  49, 17, 57, 25 };
    string cipher
        = bin2hex(permute(combine, final_perm, 64));
    return cipher;
}
int main()
{
    string pt, key;
    cout<<"Enter plain text(in hexadecimal)(123456ABCD132536): ";
    cin>>pt;
    cout<<"Enter key(in hexadecimal)(AABB09182736CCDD): ";
    cin>>key;
    int keyp[56]
        = { 57, 49, 41, 33, 25, 17, 9,  1,  58, 50, 42, 34,
            26, 18, 10, 2,  59, 51, 43, 35, 27, 19, 11, 3,
            60, 52, 44, 36, 63, 55, 47, 39, 31, 23, 15, 7,
            62, 54, 46, 38, 30, 22, 14, 6,  61, 53, 45, 37,
            29, 21, 13, 5,  28, 20, 12, 4 };
    key = permute(key, keyp, 56);
    int shift_table[16] = { 1, 1, 2, 2, 2, 2, 2, 2,
                            1, 2, 2, 2, 2, 2, 2, 1 };
    int key_comp[48] = { 14, 17, 11, 24, 1,  5,  3,  28,
                         15, 6,  21, 10, 23, 19, 12, 4,
                         26, 8,  16, 7,  27, 20, 13, 2,
                         41, 52, 31, 37, 47, 55, 30, 40,
                         51, 45, 33, 48, 44, 49, 39, 56,
                         34, 53, 46, 42, 50, 36, 29, 32 };
    string left = key.substr(0, 28);
    string right = key.substr(28, 28);

    vector<string> rkb;
    vector<string> rk;
    for (int i = 0; i < 16; i++) {
        left = shift_left(left, shift_table[i]);
        right = shift_left(right, shift_table[i]);
        string combine = left + right;
        string RoundKey = permute(combine, key_comp, 48);

        rkb.push_back(RoundKey);
        rk.push_back(bin2hex(RoundKey));
    }
    string cipher = encrypt(pt, rkb, rk);
    cout << "\nCipher Text: " << cipher << endl;

    reverse(rkb.begin(), rkb.end());
    reverse(rk.begin(), rk.end());
    string text = encrypt(cipher, rkb, rk);
    cout << "\nPlain Text: " << text << endl;
}
```

Output:

```
Enter plain text(in hexadecimal)(123456ABCD132536):
123456ABCD132536

Enter key(in hexadecimal)(AABB09182736CCDD):
AABB09182736CCDD

Cipher Text: E2789D9ADE6C1A3B

Plain Text: 123456ABCD132536
```

WEEK-4: RSA

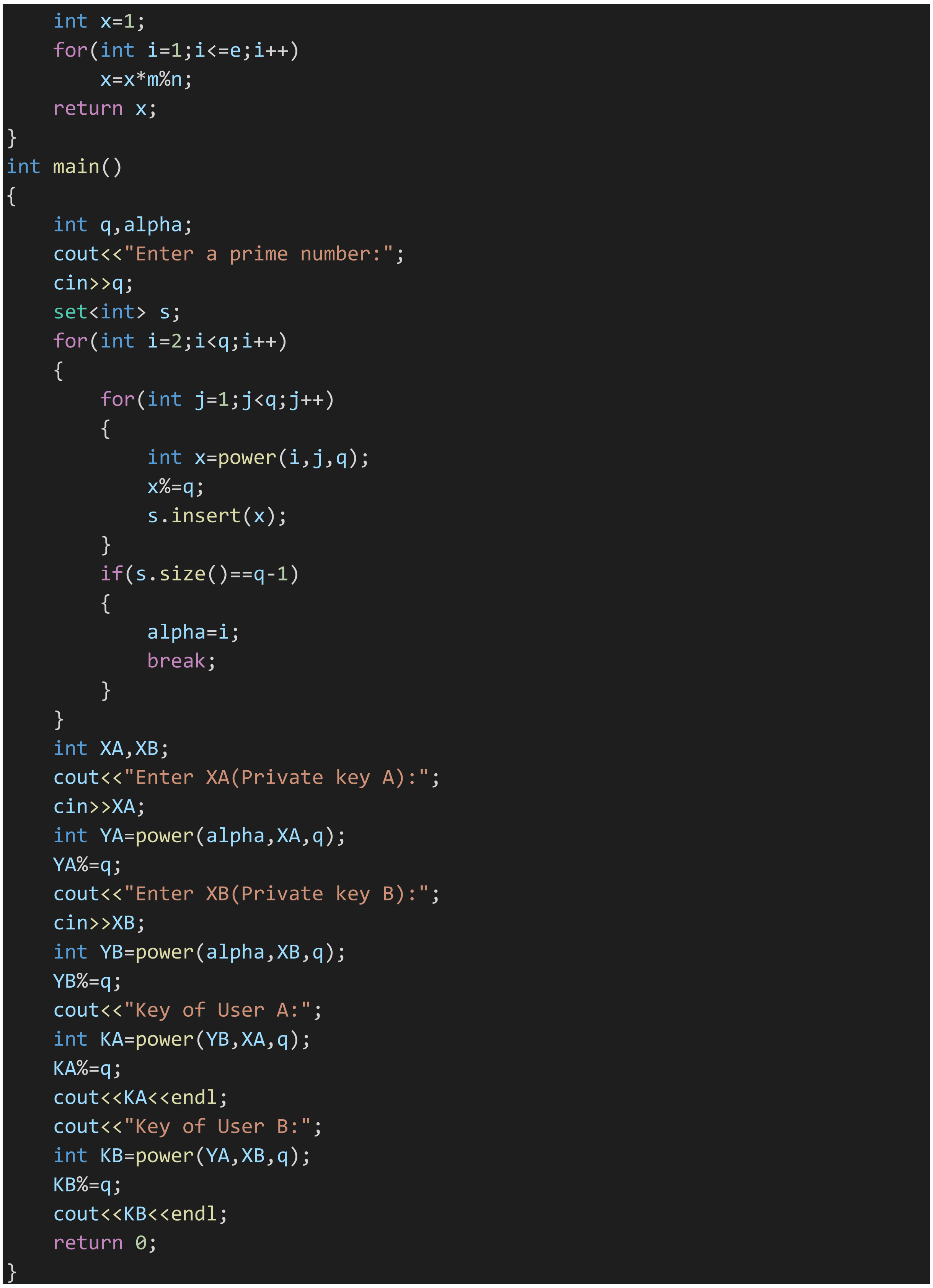
```
#include<bits/stdc++.h>
using namespace std;
int GCD(int a,int b)
{
    if(a==0)
        return b;
    else if(b==0)
        return a;
    else
        return GCD(b,a%b);
}
int power(int m,int e,int n)
{
    int x=1;
    for(int i=1;i<=e;i++)
        x=x*m%n;
    return x;
}
int main()
{
    int p,q,m;
    cout<<"Enter P value:";
    cin>>p;
    cout<<"Enter Q value:";
    cin>>q;
    cout<<"Enter M value:";
    cin>>m;
    int n=p*q;
    int e=0;
    int phi_n=(p-1)*(q-1);
    for(int i=2;i<phi_n;i++)
    {
        if(GCD(i,phi_n)==1)
        {
            e=i;
            break;
        }
    }
    int d=1;
    for(int i=2;(e*i)%phi_n!=1;i++)
    {
        d=i;
    }
    d++;
    int cipher_text=power(m,e,n)%n;
    cout<<"Cipher text is:"<<cipher_text;
    int decrypted_text=power(cipher_text,d,n)%n;
    cout<<"\nDecrypted text is:"<<decrypted_text;
    return 0;
}
```

Output:

```
Enter P value:3

Enter Q value:5

Enter M value:4

Cipher text is:4

Decrypted text is:4
```

WEEK-5: Deffie Helman

```
#include<bits/stdc++.h>
using namespace std;
int power(int m,int e,int n)
{
```



```
    int x=1;
    for(int i=1;i<=e;i++)
        x=x*m%n;
    return x;
}
int main()
{
    int q,alpha;
    cout<<"Enter a prime number:";
    cin>>q;
    set<int> s;
    for(int i=2;i<q;i++)
    {
        for(int j=1;j<q;j++)
        {
            int x=power(i,j,q);
            x%=q;
            s.insert(x);
        }
        if(s.size()==q-1)
        {
            alpha=i;
            break;
        }
    }
    int XA,XB;
    cout<<"Enter XA(Private key A):";
    cin>>XA;
    int YA=power(alpha,XA,q);
    YA%=q;
    cout<<"Enter XB(Private key B):";
    cin>>XB;
    int YB=power(alpha,XB,q);
    YB%=q;
    cout<<"Key of User A:";
    int KA=power(YB,XA,q);
    KA%=q;
    cout<<KA<<endl;
    cout<<"Key of User B:";
    int KB=power(YA,XB,q);
    KB%=q;
    cout<<KB<<endl;
    return 0;
}
```

Output:

```
Enter a prime number:11

Enter XA(Private key A):8

Enter XB(Private key B):4

Key of User A:4

Key of User B:4
```