

PRACTICAL NO.: 6

CIPHERING

Date: 08/03/21

AIM: Write a C Program to implement Cipher text using classical Cipher method.

THEORY:

A cipher (or cypher) is an algorithm for performing encryption or decryption - a series of well-defined steps that can be followed as a procedure. An alternative, less common term is encipherment. When using a cipher, the original information is known as plaintext, and the encrypted form as ciphertext. The ciphertext message contains all the information of the plaintext message, but is not in a format readable by a human or computer without the proper mechanism to decrypt it; it should resemble random gibberish to those not intended to read it.

Classical ciphers are often divided into transposition ciphers and substitution ciphers.

Most modern ciphers can be categorized in several ways:

1. By whether they work on blocks of symbols usually of a fixed size (block ciphers), or on a continuous stream of symbols (stream ciphers).
2. By whether the same key is used for both encryption and decryption (symmetric key algorithms), or if a different key is used for each (asymmetric key algorithms). If the algorithm is symmetric, the key must be known to the recipient and sender and to no one else. If the algorithm is an asymmetric one, the enciphering key is different from, but closely related to, the deciphering key. If one key cannot be deduced from the other, the asymmetric key algorithm has the public/private key property and one of the keys may be made public without loss of confidentiality.

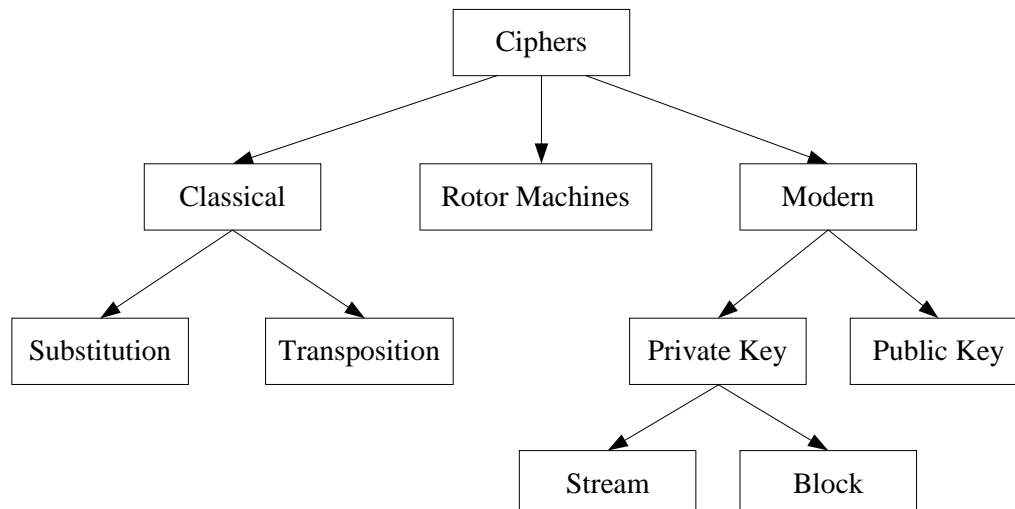


Figure 1: Types of Ciphers

Substitution Cipher

In a substitution cipher, each letter or group of letters is replaced by another letter or group of letters to disguise it. One of the oldest known ciphers is the Caesar cipher, attributed to Julius Caesar. In this method, *a* becomes *D*, *b* becomes *E*, *c* becomes *F*, ... , and *z* becomes *C*. For example, *attack* becomes *DWWDFN*. In examples, plaintext will be given in lower case letters, and ciphertext in upper case letters.

A slight generalization of the Caesar cipher allows the ciphertext alphabet to be shifted by *k* letters, instead of always 3. In this case *k* becomes a key to the general method of circularly shifted alphabets. The Caesar cipher may have fooled Pompey, but it has not fooled anyone since.

The next improvement is to have each of the symbols in the plaintext, say, the 26 letters for simplicity, map onto some other letter. For example,

Plaintext: a b c d e f g h i j k l m n o p q r s t u v w x y z

Ciphertext: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

The general system of symbol-for-symbol substitution is called a monoalphabetic substitution, with the key being the 26-letter string corresponding to the full alphabet. For the key above, the plaintext *attack* would be transformed into the ciphertext *QZZQEA*.

Transposition Ciphers

Substitution ciphers preserve the order of the plaintext symbols but disguise them. Transposition ciphers, in contrast, reorder the letters but do not disguise them. Figure 2 depicts a common transposition cipher, the columnar transposition. The cipher is keyed by a word or phrase not containing any repeated letters. In this example, MEGABUCK is the key.

The purpose of the key is to number the columns, column 1 being under the key letter closest to the start of the alphabet, and so on. The plaintext is written horizontally, in rows, padded to fill the matrix if need be. The ciphertext is read out by columns, starting with the column whose key letter is the lowest.

<u>M</u>	<u>E</u>	<u>G</u>	<u>A</u>	<u>B</u>	<u>U</u>	<u>C</u>	<u>K</u>
<u>7</u>	<u>4</u>	<u>5</u>	<u>1</u>	<u>2</u>	<u>8</u>	<u>3</u>	<u>6</u>
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

Plaintext

pleasetransferonemilliondollarsto
myswissbankaccountsixtwo

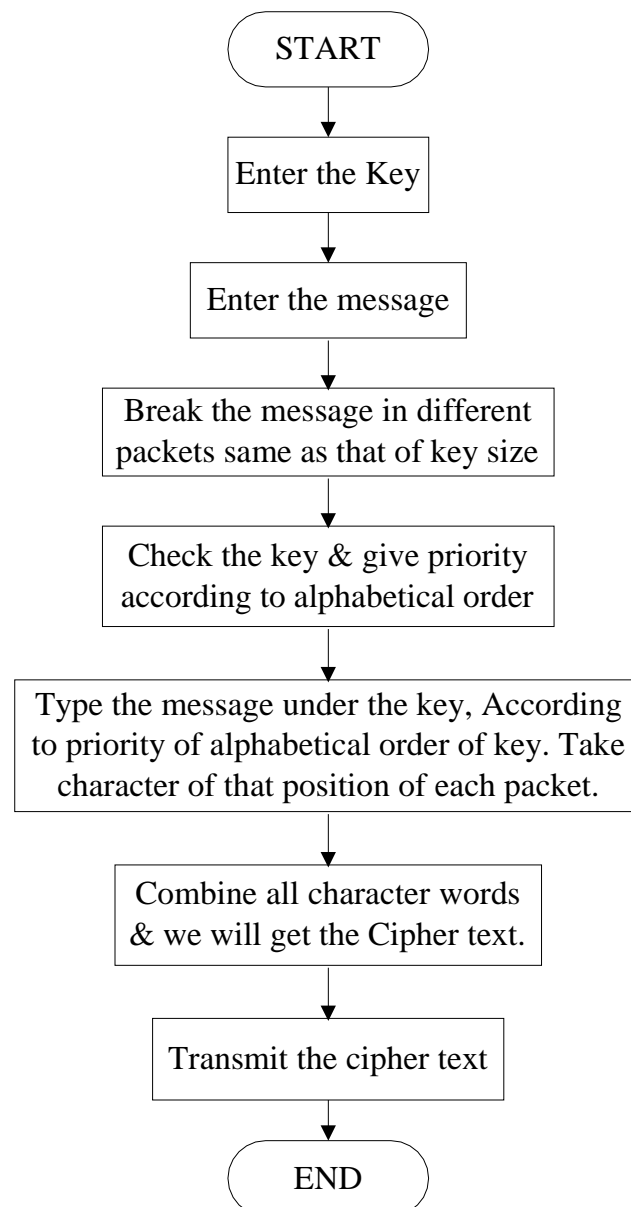
Ciphertext

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUEIRICXB

Figure 2: Transposition Cipher

ALGORITHM:

Algorithm for transposition cipher is:



C++ PROGRAM:**Part 1:**

```
string encryption(string message, int k)
{
    for (int i = 0; i < sz(message); i++)
    {
        int c = message[i] - 'a';
        c += k;
        c %= 26;
        message[i] = (char)(c + 'a');
    }
    return message;
}

string decryption(string message, int k)
{
    for (int i = 0; i < sz(message); i++)
    {
        int c = message[i] - 'a';
        c -= k;
        int val = ceil(abs(c) / (double)26);
        c += 26 * val;
        c %= 26;
        message[i] = (char)(c + 'a');
    }
    return message;
}

int32_t main()
{
    string message;
    int key;
    cin >> message >> key;
    string encryptedData = encryption(message, key);
    cout << "Encrypted: " << encryptedData << endl;
    string decryptedData = decryption(encryptedData, key);
    cout << "Decrypted: " << decryptedData << endl
}
```

```

ip1.txt x
1 sldvbsdvbsdacbskdjdc
2 3

op1.txt x
1 Encrypted: vogyevgyevgdfevngmgfe
2 Decrypted: sldvbsdvbsdacbskdjdc
3

```

```

ip1.txt x
1 snasjnasckjasniasnaas
2 7777

op1.txt x
1 Encrypted: vqdvmdvfnmdvqldvqddv
2 Decrypted: snasjnasckjasniasnaas
3

```

Part 2:

```

string enc, dyc;
string encryption(string message) {
    for (int i = 0; i < message.size(); i++) {
        message[i] = enc[message[i] - 'a'];
    }
    return message;
}

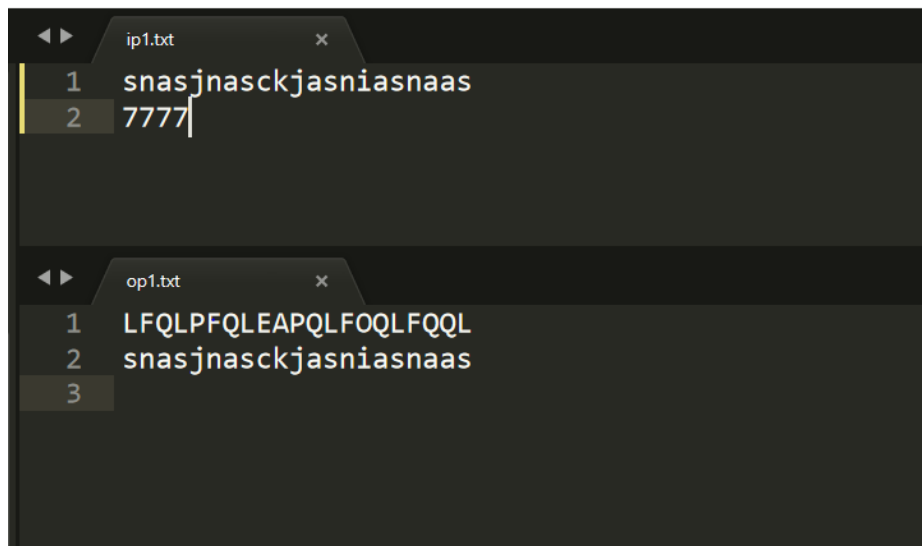
string decryption(string encrypted) {
    for (int i = 0; i < encrypted.size(); i++) {
        encrypted[i] = dyc[encrypted[i] - 'A'];
    }
    return encrypted;
}

```

```

int main()
{
    KruNULL();
    enc = "QWERTYUIOPASDFGHJKLZXCVBNM";
    for (int i = 0; i < 26; i++) {
        dyc[enc[i] - 'A'] = i + 'a';
    }
    string message;
    cin >> message;
    string encryptedData = encryption(message);
    cout << encryptedData << endl;
    string decryptedData = decryption(encryptedData);
    cout << decryptedData << endl;
}

```



Part 3:

```

string key, msg;
map<int, int> keyMap;

void setPermutationOrder()
{
    for (int i = 0; i < key.length(); i++)
    {
        keyMap[key[i]] = i;
    }
}

string encryptMessage()
{
    int row, col, j;
    string cipher = "";
    col = key.length();
    row = msg.length() / col;
    if (msg.length() % col)
        row += 1;
}

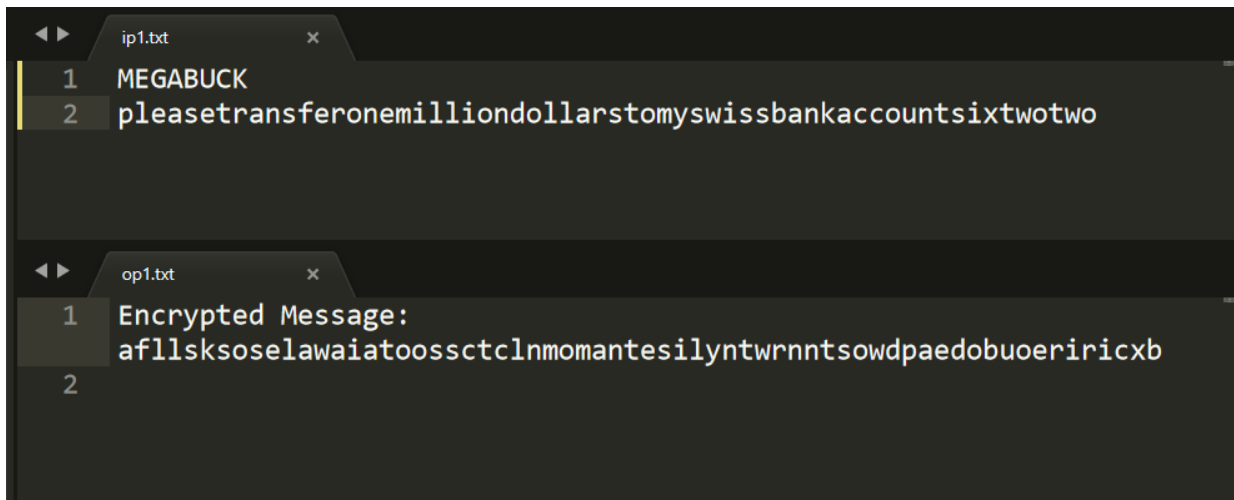
```

```

    char matrix[row][col];
    int idx = 0;
    for (int i = 0, k = 0; i < row; i++)
    {
        for (int j = 0; j < col; )
        {
            if (msg[k] == '\\0')
            {
                matrix[i][j] = idx + 'a';
                idx++;
                j++;
            }
            if ( isalpha(msg[k]) || msg[k] == ' ')
            {
                matrix[i][j] = msg[k];
                j++;
            }
            k++;
        }
    }
    for (auto ii : keyMap)
    {
        j = ii.second;
        for (int i = 0; i < row; i++)
        {
            if ( isalpha(matrix[i][j]) || matrix[i][j] == ' '
|| matrix[i][j] == '_')
                cipher += matrix[i][j];
        }
    }
    return cipher;
}

int32_t main()
{
    cin >> key >> msg;
    setPermutationOrder();
    string encrypted_message = encryptMessage();
    cout << "Encrypted Message: " << encrypted_message << endl;
}

```



Part 4: Playfair

```
void removeSpaces(string &s) {
    auto it = s.begin();
    while (it != s.end()) {
        *it == ' ' ? s.erase(it) : it++;
    }
}

void lowerCase(string &s) {
    for (int i = 0; i < s.size(); i++) {
        if (s[i] >= 'A' && s[i] <= 'Z') {
            s[i] += ('a' - 'A');
        }
    }
}

void evenString(string &s) {
    if (s.length() % 2 != 0) {
        s += 'z';
    }
}

void generate(string &key, string* keyTable) {
    unordered_set<char> st(key.begin(), key.end());
    if (st.find('j') == st.end()) {
        st.insert('j');
    }
    char x = 'a';
    for (int i = 0, k = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            if (k < key.size()) {
                keyTable[i][j] = key[k++];
            }
            else {
                while (st.find(x) != st.end()) {
                    x++;
                }
                keyTable[i][j] = x;
                x++;
            }
        }
    }
}
```



```

        }
    }
}

vector<int> search(string* table, char a, char b) {
    vector<int> ans(4);
    if (a == 'j')
        a = 'i';
    if (b == 'j')
        b = 'i';

    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            if (table[i][j] == a) {
                ans[0] = i;
                ans[1] = j;
            }
            else if (table[i][j] == b) {
                ans[2] = i;
                ans[3] = j;
            }
        }
    }
    return ans;
}

void encrypt(string &s, string* keyTable) {
    for (int i = 0; i < s.size(); i = i + 2) {
        vector<int> ans = search(keyTable, s[i], s[i + 1]);

        if (ans[0] == ans[2]) {
            s[i] = keyTable[ans[0]][(ans[1] + 1) % 5];
            s[i + 1] = keyTable[ans[0]][(ans[3] + 1) % 5];
        }
        else if (ans[1] == ans[3]) {
            s[i] = keyTable[(ans[0] + 1) % 5][ans[1]];
            s[i + 1] = keyTable[(ans[2] + 1) % 5][ans[1]];
        }
        else {
            s[i] = keyTable[ans[0]][ans[3]];
            s[i + 1] = keyTable[ans[2]][ans[1]];
        }
    }
}

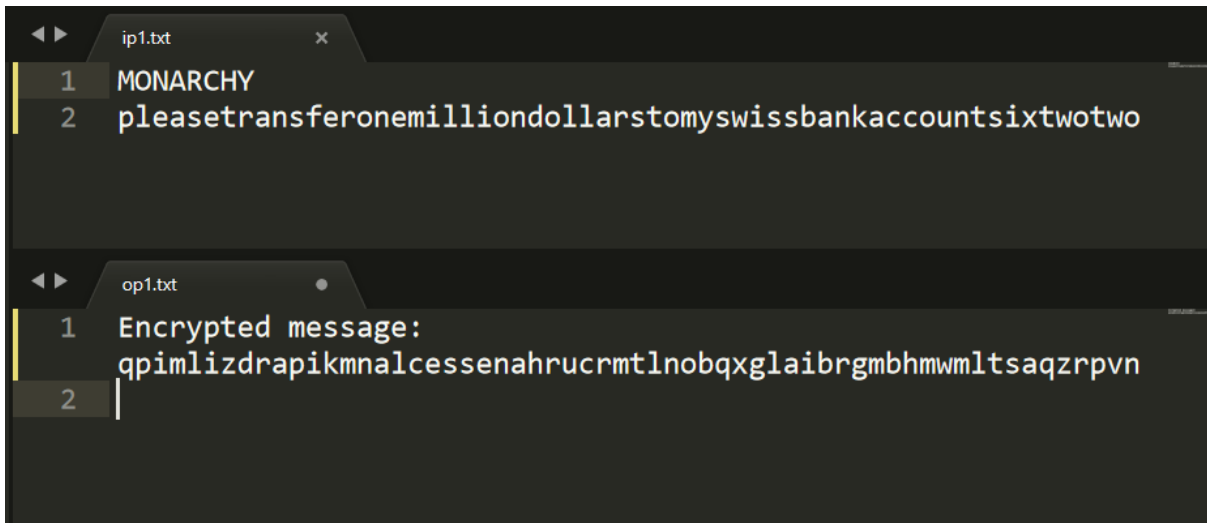
void encryption(string &s, string &key) {
    removeSpaces(s);
    lowerCase(s);
    evenString(s);

    removeSpaces(key);
    lowerCase(key);

    string keyTable[5];
    generate(key, keyTable);
}

```

```
        encrypt(s, keyTable);  
    }  
int32_t main()  
{  
    string key, s;  
    cin >> key >> s;  
    encryption(s, key);  
    cout << "Encrypted message: " << s << endl;  
}
```



The screenshot shows a code editor with two tabs: 'ip1.txt' and 'op1.txt'. The 'ip1.txt' tab is active and shows two lines of input text: '1 MONARCHY' and '2 pleasetransferonemilliondollarstomyswissbankaccountsixtwo'. The 'op1.txt' tab is also visible and shows the output: '1 Encrypted message:' followed by the encrypted text 'qpimlizdrapikmnlcessenahrucrmtnobqxglabrgmbhmwmmltsaqzrpvn' on the next line. Line 2 of 'op1.txt' is currently empty.

CONCLUSION:

In this experiment, we implemented the code for cipher text and understood all the algorithms for various types of cipher with the help of desired value of key and flowchart. The output of the codes were consistent with the desired result.

Signature of the Teacher