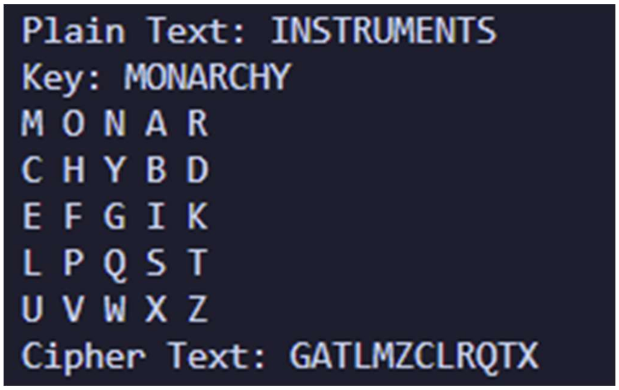
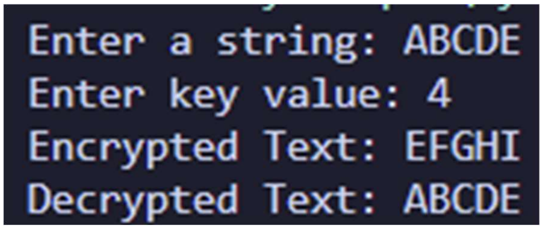
```cpp
#include <iostream>

#include <string>

using namespace std;


string caesar_cipher_encrypt(string text, int key){

    string res = "";

    for(int i = 0; i<text.size(); i++){

        res.push_back(((text[i] - 'A' + key) % 26) + 'A');

    }

    return res;

}


string caesar_cipher_decrypt(string text, int key){

    string res = "";

    for(int i = 0; i < text.size(); i++){

        char decrypted_char = ((text[i] - 'A' - key) % 26);

        if (decrypted_char < 0) {

            decrypted_char += 26;

        }

        res.push_back(decrypted_char + 'A');

    }

    return res;

}


// caesar cipher

int main(){

    string text;

    int key;

    cout<<"Enter a string: ";

    cin>>text;

    cout<<"Enter key value: ";
    cin>>key;
    string encrypted = caesar_cipher_encrypt(text, key);
    cout<<"Encrypted Text: "<<encrypted<<endl;
    cout<<"Decrypted Text: "<<caesar_cipher_decrypt(encrypted, key);
    return 0;
}
```

```
#include <iostream>
#include <vector>
using namespace std;

vector<vector<char>> generate_key_square(string key, string text){
    vector<bool> filled(26, false);
    vector<vector<char>> key_square(5, vector<char>(5));
    int row = 0, col = 0;
    for(char c: key){
        if(c == 'J') c = 'I';
        if(!filled[c - 'A']) {
            key_square[row][col] = c;
            filled[c-'A'] = true;
            col++;
            if(col == 5){
                row++;
                col = 0;
            }
        }
    }

    for(int i = 0; i<26; i++){
        if(!filled[i] && i+'A' != 'J'){
            key_square[row][col] = i + 'A';
            col++;
            if(col == 5){
                row++;
                col = 0;
            }
        }
    }
    return key_square;
}

vector<pair<char, char>> get_digrams(string text, char filler){
    vector<pair<char, char>> res;
    for(int i = 0; i<text.size(); i+=2){
        if(i+1 < text.size() && text[i] == text[i+1]){
            res.push_back({text[i], filler});
            i--;
        } else if(i + 1 < text.size()){
            res.push_back({text[i], text[i+1]});
        } else {
            res.push_back({text[i], filler});
        }
    }
    return res;
}

pair<int, int> get_char_coordinates(vector<vector<char>>& square, char c){
    for(int i = 0; i<square.size(); i++){
        for(int j = 0; j<square[i].size(); j++){
            if(square[i][j] == c){
                return {i, j};
            }
        }
    }
    return {-1, -1};
}

string encrypt_playfair(vector<vector<char>>& square, vector<pair<char, char>>& digrams){
    string res;
    for(auto d: digrams){
        pair<int, int> coords_first = get_char_coordinates(square, d.first);
        pair<int, int> coords_second = get_char_coordinates(square, d.second);
        if(coords_first.second == coords_second.second) { // both in same column
            int row1 = (coords_first.first + 1) % 5;
            int col1 = coords_first.second;
            int row2 = (coords_second.first + 1) % 5;
            int col2 = coords_second.second;
            res.push_back(square[row1][col1]);
            res.push_back(square[row2][col2]);
        } else if (coords_first.first == coords_second.first) { // both in same row
            int row1 = coords_first.first;
            int col1 = (coords_first.second + 1) % 5;
            int row2 = coords_second.first;
            int col2 = (coords_second.second + 1) % 5;
            res.push_back(square[row1][col1]);
            res.push_back(square[row2][col2]);
        } else { // rectangle swap
            int row1 = coords_first.first;
            int col1 = coords_second.second;
            int row2 = coords_second.first;
            int col2 = coords_first.second;
            res.push_back(square[row1][col1]);
            res.push_back(square[row2][col2]);
        }
    }
    return res;
}

//playfair cipher
```

```
int main(){
    string key, text;
    cout<<"Plain Text: ";
    cin>>text;
    cout<<"Key: ";
    cin>>key;
    vector<vector<char>> square = generate_key_square(key, text);
    for(auto v: square){
        for(auto c: v){
            cout << c << " ";
        }
        cout << endl;
    }
    vector<pair<char, char>> digrams;
    digrams = get_digrams(text, 'Z');
    string res = encrypt_playfair(square, digrams);
    cout <<"Cipher Text: " << res << endl;
    return 0;
}
```



```
Plain Text: INSTRUMENTS
Key: MONARCHY
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z
Cipher Text: GATLMZCLRQTX
```



```
Enter a string: ABCDE
Enter key value: 4
Encrypted Text: EFGHI
Decrypted Text: ABCDE
```