```cpp

// Create Performance Task 5/02/2022

// I learned about vigenere cyphers from https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Base.html

#include <iostream>
#include <string>
#include <ctime>
#include <vector>

//defining std namespaces
using std::cout;
using std::cin;
using std::endl;
using std::getline;
using std::string;
using std::vector;

const int TABLESIZE = 96; // ascii values 32 - 128
const int KEYLENGTH = 20; // key length doesn't matter because if the length is bigger than key, it jsut wraps around


vector<string> encryptVigenere(string); // takes in plaintext string, returns cyphertext string and key string in vector
string decryptVigenere(string, string); // takes in cyphertext and key string, returns plaintext string

void createTable(char(&table)[TABLESIZE][TABLESIZE]) {
    //assign letters to table
    for (int i = 0; i < TABLESIZE; i++) {
        for (int j = 0; j < TABLESIZE; j++) {
            table[i][j] = (char)((i + j + 32) % TABLESIZE) + 32;
            // first +32 to make lowest i + j possible a spacebar(first viable character) 0->31 in ascii are unusable
            // % TABLESIZE makes values over TABLESIZE go back to 0 + overflow, then +32 makes it at least first viable character
        }
    }
}


// all three variables below only used for input
string plainText;
string cypherText;
string key;


int main() {

    string dtest;
    dtest = decryptVigenere("AW^`c", "y2244|VJkwJI%_oEJr(m");
    cout << "Test 1: " << dtest << endl;
    dtest = decryptVigenere("Z$*@IP,", "3U[|'wggq}x_@wcC'.\"c");
    cout << "Test 2: " << dtest << endl;

    //display title on start
    cout << ".-\"-.     .-\"-.     .-\"-.     .-\"-.     .-\"-.     .-\"-." << endl;
    cout << "     \"-.-\"     \"-.-\"     \"-.-\"     \"-.-\"     \"-.-\"     \"-.-\"" << endl;
    cout << "\n\t\tVigenere Encryption Program" << endl << endl;
    cout << ".-\"-.     .-\"-.     .-\"-.     .-\"-.     .-\"-.     .-\"-." << endl;
    cout << "     \"-.-\"     \"-.-\"     \"-.-\"     \"-.-\"     \"-.-\"     \"-.-\"" << endl;
    //This ASCII art was found from https://asciiart.website/index.php?art=art%20and%20design/borders

    while (true) {
        cout << endl << "/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)/)" << endl;
        cout << "(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/(/" << endl << endl;
        //This ASCII art was found from https://asciiart.website/index.php?art=art%20and%20design/borders

        //get the user input on whether to encrypt or decrypt
        string encryptOrDecryptInput;
        cout << "[e]ncrypt | [d]ecrypt\t<< ";
        getline(cin, encryptOrDecryptInput);
        if (encryptOrDecryptInput[0] == 'e' || encryptOrDecryptInput[0] == 'E') {
            cout << "input plaintext\t<< ";
            getline(cin, plainText);
            vector <string> cyphertextAndKey = encryptVigenere(plainText); // encryption function call
            cout << "cyphertext\t\t>> " << cyphertextAndKey[0] << endl;
            cout << "key\t\t\t>> " << cyphertextAndKey[1] << endl;
        }
        if (encryptOrDecryptInput[0] == 'd' || encryptOrDecryptInput[0] == 'D') {
            cout << "input cyphertext\t<< ";
            getline(cin, cypherText);
            cout << "input key\t<< ";
            getline(cin, key);
            if (key.length() == KEYLENGTH) {
                string plainTextDecrypt = decryptVigenere(cypherText, key); // decryption function call
                cout << "plaintext\t\t>> " << plainTextDecrypt << endl;
            }
            else {
                cout << "*** INVALID KEY | the key must be exactly " << KEYLENGTH << " characters long ***" << endl;
            }
        }
    }
}


vector <string> encryptVigenere(string plainText) {
    int h, i, j; // initialize once to save memory

    char table[TABLESIZE][TABLESIZE]; //create table
    createTable(table);

    srand((unsigned)time(NULL));
    //create a list of all possible characters in (char) instead of (int)
    char alphanum[95]; // excludes spacebar in key
    for (int n = 0; n < 95; n++) {
        alphanum[n] = (char)n + 33;
    }

    //generate key
    string key = "";
    for (int i = 0; i < KEYLENGTH; ++i) {
        key += alphanum[rand() % (sizeof(alphanum) - 1)];
    }

    //mod 95 returns how many letters overflow

    int lenOfPlainText = plainText.length();
    vector<char> cypherText(lenOfPlainText); // allocates enough memory for cypherText

    //this loop takes cypherText out of table by using randomly generated key
    for (h = 0; h < lenOfPlainText; h++) { // loop through each letter of plainText
        for (i = 0; i < TABLESIZE; i++) { // loop through the table a row at a time
            if (table[i][0] == plainText[h]) { // find the row that starts with the plaintext character
                for (j = 0; j < TABLESIZE; j++) { // loop through that row
                    if (table[0][j] == key[h % key.length()]) { // find the column that the key character is in
                        cypherText[h] = table[i][j]; // add the character at [plaintext row][key column] to cypherText
                    }
                }
            }
        }
    }

    string cypherTextStr(cypherText.begin(), cypherText.end());
    vector<string> cyphertextAndKey{ cypherTextStr, key }; // puts the cypherText and key into a vector in order to return both
    return cyphertextAndKey;
```

```cpp
}


string decryptVigenere(string cypherText, string key) {
    int h, i, j; // initialize once to save memory

    char table[TABLESIZE][TABLESIZE];
    createTable(table);

    int lenOfCypherText = cypherText.length();
    vector<char> plainTextDecrypt(lenOfCypherText); // allocates enough memory for plainTextDecrypt

    //use cypherText and key to take plainText out of table
    for (h = 0; h < lenOfCypherText; h++) { // loop through each letter of cypherText
        for (i = 0; i < TABLESIZE; i++) { // loop through the table a row at a time
            if (table[i][0] == key[h % key.length()]) { // find the row that starts with the key character
                for (j = 0; j < TABLESIZE; j++) { // loop through that row
                    if (table[i][j] == cypherText[h]) { // find the column of that row that the cypherText is in
                        plainTextDecrypt[h] = table[0][j]; // add the character at [first row][column cypherText character is in when start of row=key]
                    }
                }
            }
        }
    }

    string plainTextDecryptStr(plainTextDecrypt.begin(), plainTextDecrypt.end());
    return plainTextDecryptStr;
}
```