#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;

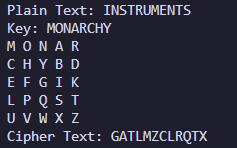
    digrams *=* *get\_digrams*(text, 'Z');

*string* res = *encrypt\_playfair*(square, digrams);

    cout *<<*"Cipher Text: " *<<* res *<<* *endl*;

    return 0;

}



A screenshot of a computer

Description automatically generated