**Practical – 1**

**Aim :- Write down a program to implement Caesar Cipher Encryption.**

#include <stdio.h>

int main()

{

    char message[100], ch;

    int i, key;

    printf("\nEnter a message to encrypt: ");

    gets(message);

    printf("Enter key: ");

    scanf("%d", &key);

    for (i = 0; message[i] != '\0'; ++i)

    {

        ch = message[i];

        if (ch >= 'a' && ch <= 'z')

        {

            ch = ch + key;

            if (ch > 'z')

            {

                ch = ch - 'z' + 'a' - 1;

            }

            message[i] = ch;

        }

        else if (ch >= 'A' && ch <= 'Z')

        {

            ch = ch + key;

            if (ch > 'Z')

            {

                ch = ch - 'Z' + 'A' - 1;

            }

            message[i] = ch;

        }

    }

    printf("\nEncrypted message: %s\n", message);

    for (i = 0; message[i] != '\0'; ++i)

    {

        ch = message[i];

        if (ch >= 'a' && ch <= 'z')

        {

            ch = ch - key;

            if (ch < 'a')

            {

                ch = ch + 'z' - 'a' + 1;

            }

            message[i] = ch;

        }

        else if (ch >= 'A' && ch <= 'Z')

        {

            ch = ch - key;

            if (ch < 'A')

            {

                ch = ch + 'Z' - 'A' + 1;

            }

            message[i] = ch;

        }

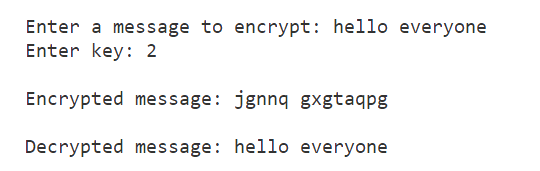
    }

    printf("\nDecrypted message: %s\n\n", message);

    return 0;

}

**Output:**



**Practical – 2**

**Aim :- W.A.P. to implement Affine Cipher with equation c=3x+12.**

#include <iostream>

#include <string.h>

#include <stdlib.h>

using namespace std;

string encryptionMessage(string Msg)

{

    string CTxt = "";

    int a = 3;

    int b = 12;

    for (int i = 0; i < Msg.length(); i++)

    {

        CTxt = CTxt + (char)((((a \* Msg[i]) + b) % 26) + 65);

    }

    return CTxt;

}

string decryptionMessage(string CTxt)

{

    string Msg = "";

    int a = 3;

    int b = 12;

    int a\_inv = 0;

    int flag = 0;

    for (int i = 0; i < 26; i++)

    {

        flag = (a \* i) % 26;

        if (flag == 1)

        {

            a\_inv = i;

        }

    }

    for (int i = 0; i < CTxt.length(); i++)

    {

        Msg = Msg + (char)(((a\_inv \* ((CTxt[i] - b)) % 26)) + 65);

    }

    return Msg;

}

int main(int argc, char \*\*argv)

{

    cout << "\nEnter the message: ";

    string message;

    cin >> message;

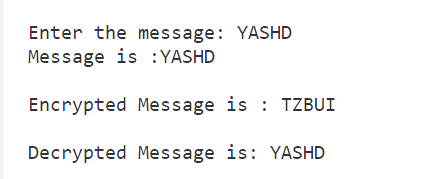
    cout << "Message is :" << message;

    cout << "\n\nEncrypted Message is : " << encryptionMessage(message);

    cout << "\n\nDecrypted Message is: " << decryptionMessage(encryptionMessage(message)) << endl << endl;

}

**Ouput:**



**Practical – 3**

**Aim : W.A.P. to implement Playfair Cipher with key ldrp**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE 30

void toLowerCase(char plain[], int ps)

{

    int i;

    for (i = 0; i < ps; i++)

    {

        if (plain[i] > 64 && plain[i] < 91)

            plain[i] += 32;

    }

}

int removeSpaces(char \*plain, int ps)

{

    int i, count = 0;

    for (i = 0; i < ps; i++)

        if (plain[i] != ' ')

            plain[count++] = plain[i];

    plain[count] = '\0';

    return count;

}

void generateKeyTable(char key[], int ks, char keyT[5][5])

{

    int i, j, k, flag = 0, \*dicty;

    dicty = (int \*)calloc(26, sizeof(int));

    for (i = 0; i < ks; i++)

    {

        if (key[i] != 'j')

            dicty[key[i] - 97] = 2;

    }

    dicty['j' - 97] = 1;

    i = 0;

    j = 0;

    for (k = 0; k < ks; k++)

    {

        if (dicty[key[k] - 97] == 2)

        {

            dicty[key[k] - 97] -= 1;

            keyT[i][j] = key[k];

            j++;

            if (j == 5)

            {

                i++;

                j = 0;

            }

        }

    }

    for (k = 0; k < 26; k++)

    {

        if (dicty[k] == 0)

        {

            keyT[i][j] = (char)(k + 97);

            j++;

            if (j == 5)

            {

                i++;

                j = 0;

            }

        }

    }

}

void search(char keyT[5][5], char a, char b, int arr[])

{

    int i, j;

    if (a == 'j')

        a = 'i';

    else if (b == 'j')

        b = 'i';

    for (i = 0; i < 5; i++)

    {

        for (j = 0; j < 5; j++)

        {

            if (keyT[i][j] == a)

            {

                arr[0] = i;

                arr[1] = j;

            }

            else if (keyT[i][j] == b)

            {

                arr[2] = i;

                arr[3] = j;

            }

        }

    }

}

int mod5(int a) { return (a % 5); }

int prepare(char str[], int ptrs)

{

    if (ptrs % 2 != 0)

    {

        str[ptrs++] = 'z';

        str[ptrs] = '\0';

    }

    return ptrs;

}

void encrypt(char str[], char keyT[5][5], int ps)

{

    int i, a[4];

    for (i = 0; i < ps; i += 2)

    {

        search(keyT, str[i], str[i + 1], a);

        if (a[0] == a[2])

        {

            str[i] = keyT[a[0]][mod5(a[1] + 1)];

            str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];

        }

        else if (a[1] == a[3])

        {

            str[i] = keyT[mod5(a[0] + 1)][a[1]];

            str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];

        }

        else

        {

            str[i] = keyT[a[0]][a[3]];

            str[i + 1] = keyT[a[2]][a[1]];

        }

    }

}

void encryptByPlayfairCipher(char str[], char key[])

{

    char ps, ks, keyT[5][5];

    ks = strlen(key);

    ks = removeSpaces(key, ks);

    toLowerCase(key, ks);

    ps = strlen(str);

    toLowerCase(str, ps);

    ps = removeSpaces(str, ps);

    ps = prepare(str, ps);

    generateKeyTable(key, ks, keyT);

    encrypt(str, keyT, ps);

}

int main()

{

    char str[SIZE], key[SIZE];

    strcpy(key, "ldrp");

    printf("\n\nKey text: %s\n", key);

    strcpy(str, "Ksv is a university");

    printf("Plain text: %s\n\n", str);

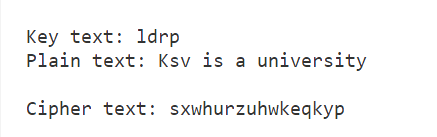
    encryptByPlayfairCipher(str, key);

    printf("Cipher text: %s\n\n", str);

    return 0;

}

**Output :**



**Practical – 4**

**Aim : W.A.P. to implement polyalphabetic Cipher.**

#include <stdio.h>

#include <conio.h>

#include <string.h>

void main()

{

    char msg[30], key[30], k[20], ct[20], pt[20];

    int lenm, lenk, i, j;

    printf("\nEnter Message : ");

    gets(msg);

    printf("Enter Key : ");

    gets(key);

    lenm = strlen(msg);

    lenk = strlen(key);

    for (i = 0; i < lenm; i++, j++)

    {

        if (j == lenk)

            j = 0;

        k[i] = key[j];

    }

    for (i = 0; i < lenm; i++)

    {

        ct[i] = ((msg[i] + k[i]) % 26) + 'A';

    }

    ct[i] = '\0';

    for (i = 0; i < lenm; i++)

    {

        pt[i] = (((ct[i] - k[i]) + 26) % 26) + 'A';

    }

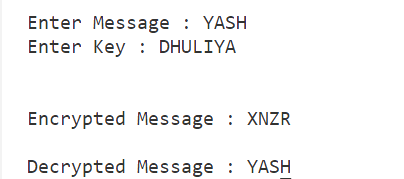
    pt[i] = '\0';

    printf("\n\nEncrypted Message : %s", ct);

    printf("\n\nDecrypted Message : %s\n", pt);

}

**Output:**

****

**Practical – 5**

**Aim : W.A.P. to implement AutoKey.**

public class p5 {

    private static final String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    public static void main(String[] args) {

        String msg = "DYASH";

        String key = "HII";

        if (key.matches("[-+]?\\d\*\\.?\\d+"))

            key = "" + alphabet.charAt(Integer.parseInt(key));

        String enc = autoEncryption(msg, key);

        System.out.println("\nPlaintext : " + msg);

        System.out.println("\nEncrypted : " + enc);

        System.out.println("\nDecrypted : " + autoDecryption(enc, key) + "\n");

    }

    public static String autoEncryption(String msg, String key) {

        int len = msg.length();

        String newKey = key.concat(msg);

        newKey = newKey.substring(0, newKey.length() - key.length());

        String encryptMsg = "";

        for (int x = 0; x < len; x++) {

            int first = alphabet.indexOf(msg.charAt(x));

            int second = alphabet.indexOf(newKey.charAt(x));

            int total = (first + second) % 26;

            encryptMsg += alphabet.charAt(total);

        }

        return encryptMsg;

    }

    public static String autoDecryption(String msg, String key) {

        String currentKey = key;

        String decryptMsg = "";

        // applying decryption algorithm

        for (int x = 0; x < msg.length(); x++) {

            int get1 = alphabet.indexOf(msg.charAt(x));

            int get2 = alphabet.indexOf(currentKey.charAt(x));

            int total = (get1 - get2) % 26;

            total = (total < 0) ? total + 26 : total;

            decryptMsg += alphabet.charAt(total);

            currentKey += alphabet.charAt(total);

        }

        return decryptMsg;

    }

}

**Output:**

