TASK 1

a)

The following is a implementation of the Decryption part of Caesar Cipher that takes a key and a ciphertext as inputs and provides a plaintext as output -

```cpp
#include <iostream>
#include <string>

using namespace std;

string caeser(const string& cypherText, int key) {
    string plainText = "";
    for (char ch : cypherText) {
        if (ch == ' ' || ch == '\n' ||  ch == '.') {
            plainText += ch;
            continue;
        }

        int c = (ch - key - 'A' + 26) % 26;
        plainText += static_cast<char>(c + 'A');
    }
    return plainText;
}

int main() {
    string cypherText;
    int key;

    cout << "Enter cypher text: ";
    getline(cin, cypherText);

    cout << "Enter key: ";
    cin >> key;

    string plainText = caeser(cypherText, key);
    cout << "Decrypted text: " << plainText << '\n';

    return 0;
}
```

b)

To decrypt a Caesar cipher, the shift value used for the encryption needs to be determined. Caesar ciphers involve shifting the letters of the alphabet by a certain number of positions. Since the English alphabet has 26 letters, there are only 25 possible shift values.

To decrypt the given encrypted message -

XNZ XVMIDQVG VO DDXO WPDGYDIB NPNO DN BJDIB OJ WZ BMZVO VBVDI

I used trial and error by trying different shift values until I found the correct one.

```
Enter cypher text: XNZ XVMIDQVG VO DDXO WPDGYDIB NPNO DN BJDIB OJ WZ BMZVO VBVDL
For key value 0 Decrypted text: XNZ XVMIDQVG VO DDXO WPDGYDIB NPNO DN BJDIB OJ WZ
BMZVO VBVDL
For key value 1 Decrypted text: WMY WULHCPUF UN CCWN VOCFXCHA MOMN CM AICHA NI VY
ALYUN UAUCK
For key value 2 Decrypted text: VLX VTKGBOTE TM BBVM UNBEWBGZ LNLM BL ZHBGZ MH UX
ZKXTM TZTBJ
For key value 3 Decrypted text: UKW USJFANSD SL AAUL TMADVAFY KMKL AK YGAFY LG TW
YJWSL SYSAI
For key value 4 Decrypted text: TJV TRIEZMRC RK ZZTK SLZCUZEX JLJK ZJ XFZEX KF SV
XIVRK RXRZH
For key value 5 Decrypted text: SIU SQHDYLQB QJ YYSJ RKYBTYDW IKIJ YI WEYDW JE RU
WHUQJ QWQYG
For key value 6 Decrypted text: RHT RPGCXKPA PI XXRI QJXASXCV HJHI XH VDXCV ID QT
VGTPI PVPXF
For key value 7 Decrypted text: QGS QOFBWJOZ OH WWQH PIWZRWBU GIGH WG UCWBU HC PS
UFSOH OUOWE
For key value 8 Decrypted text: PFR PNEAVINY NG VVPG OHVYQVAT FHFG VF TBVAT GB OR
TERNG NTNVD
For key value 9 Decrypted text: OEQ OMDZUHMX MF UUOF NGUXPUZS EGEF UE SAUZS FA NQ
SDQMF MSMUC
For key value 10 Decrypted text: NDP NLCYTGLW LE TTNE MFTWOTYR DFDE TD RZTYR EZ MP
RCPLE LRLTB
For key value 11 Decrypted text: MCO MKBXSFKV KD SSMD LESVNSXQ CECD SC QYSXQ DY LO
QBOKD KQKSA
```

```
For key value 12 Decrypted text: LBN LJAWREJU JC RRLC KDRUMRWP BDBC RB PXRWP CX KN
PANJC JPJRZ
For key value 13 Decrypted text: KAM KIZVQDIT IB QQKB JCQTLQVO ACAB QA OWQVO BW JM
OZMIB IOIQY
For key value 14 Decrypted text: JZL JHYUPCHS HA PPJA IBPSKPUN ZBZA PZ NVPUN AV IL
NYLHA HNHPX
For key value 15 Decrypted text: IYK IGXTOBGR GZ OOIZ HAORJOTM YAYZ OY MUOTM ZU HK
MXKGZ GMGOW
For key value 16 Decrypted text: HXJ HFWSNAFQ FY NNHY GZNQINSL XZXY NX LTNSL YT GJ
LWJFY FLFNV
For key value 17 Decrypted text: GWI GEVRMZEP EX MMGX FYMPHMRK WYWX MW KSMRK XS FI
KVIEX EKEMU
For key value 18 Decrypted text: FVH FDUQLYDO DW LLFW EXLOGLQJ VXVW LV JRLQJ WR EH
JUHDW DJDLT
For key value 19 Decrypted text: EUG ECTPKXCN CV KKEV DWKNFKPI UWUV KU IQKPI VQ DG
ITGCV CICKS
For key value 20 Decrypted text: DTF DBSOJWBM BU JJDU CVJMEJOH TVTU JT HPJOH UP CF
HSFBU BHBJR
For key value 21 Decrypted text: CSE CARNIVAL AT IICT BUILDING SUST IS GOING TO BE
GREAT AGAIQ
For key value 22 Decrypted text: BRD BZQMHUZK ZS HHBS ATHKCHMF RTRS HR FNHMF SN AD
FQDZS ZFZHP
For key value 23 Decrypted text: AQC AYPLGTYJ YR GGAR ZSGJBGLE QSQR GQ EMGLE RM ZC
EPCYR YEYGO
For key value 24 Decrypted text: ZPB ZXOKFSXI XQ FFZQ YRFIAFKD PRPQ FP DLFKD QL YB
DOBXQ XDXFN
For key value 25 Decrypted text: YOA YWNJERWH WP EEYP XQEHZEJC OQOP EO CKEJC PK XA
CNAWP WCWEM
```

It appears that a shift value of 21 is required here. Only then a meaningful sentence is formed, which is -

CSE CARNIVAL AT IICT BUILDING SUST IS GOING TO BE GREAT AGAIN

c)

In this case, a shift value of 3 is used to decrypt the encrypted message shown by the following cipher blocks.

ZKDW GR BRX JHW ZKHQ BRX FURVV D VQRZPDQ ZLWK D YDPSLUH IURVWELWH

Z	K	D	W		G	R		B	R	X		J	H	W		Z	K	H	Q

B	R	X		F	U	R	V	V		D		V	Q	R	Z	P	D	Q	

Z	L	W	K		D		Y	D	P	S	L	U	H	?		I	U	R	V

W	E	L	W	H

And the corresponding decrypted message is -

WHAT DO YOU GET WHEN YOU CROSS A SNOWMAN WITH A VAMPIRE FROSTBITE

So the filled boxes should look like this -

W	H	A	T		D	O		Y	O	U		G	E	T		W	H	E	N
Z	K	D	W		G	R		B	R	X		J	H	W		Z	K	H	Q

Y	O	U		C	R	O	S	S		A		S	N	O	W	M	A	N	
B	R	X		F	U	R	V	V		D		V	Q	R	Z	P	D	Q	

W	I	T	H		A		V	A	M	P	I	R	E		F	R	O	S
Z	L	W	K		D		Y	D	P	S	L	U	H		I	U	R	V

T	B	I	T	E
W	E	L	W	H

TASK 2

Following is the base code implemented in python (notebook) for this task -

```
import numpy as np
from collections import Counter


def frequency(encryptedtext):
    encryptedtext = encryptedtext.lower()
    encryptedtext = ''.join(filter(str.isalpha, encryptedtext))
    frequency = Counter(encryptedtext)
    return frequency


def sub_cipher(encryptedtext, mappings):


    plaintext = ""
    for c in encryptedtext:
        if c in mappings:
            plaintext += mappings[c]
        elif c.isalpha():
```

```
            plaintext += '*'
        else:
            plaintext += c
    return plaintext
```

a.

Given encrypted message is -

gtd bsvgl vf fgedsugt dffml dkcymvsf gtmg gtd chjde ha aevdsxftvc tdycf bf gh id fgehsu aehz tmexftvcf. aevdsxf qms uvod bf gtd fgedsugt jd sddx jtds yvad udgf ghbut. vs mxxvgvhs, cdhcyd dkcedff bsvgl gtehbut yhod,amzvyl, aevdsxf, msx hgtdef ftmed fghevdf ha avsxvsu qhzzhs uehbsx jvgt fhzdhsd.gtded med zmsl idsdavgf ha fgmlvsu bsvgdx vs ghbut gvzdf, mf vg tdycf gh amqd qtmyydsuvsu fvgbmgvhsf jvgt qhbemud. gtd vzchegmsqd ha fgmlvsu bsvgdx tmf fgebqp m qthex mzhsu zmsl cdhcyd gtehbuthbg tvfghel.pddcvsu zdzhevdf ha jtmg jd tmod mqqhzcyvftdx gtehbuthbg tvfghel qms tdyc bf fdd thj vsxvovxbmyf msx qhzzbsvgvdf tmod cdefdodedx gtehbut ghbut gvzdf msx vsgh m ievutgde

Here “gtd” appears numerous times which can be nothing but “the”, so-

g=t

t=h

d=e

So now replacing with these mapping, the text becomes –

the bsvtl vf fteesuth effml ekcymvsf thmt the chjee ha aevesxfhvc heycf bf th ie ftehsu aehz hmexfhvcf. aevesxf qms uvoe bf the fteesuth je seex jhes yvae uetf thbuh. vs mxxvtvhs, cehcye ekceeff bsvtl thehbuh yhoe,amzvyl, aevesxf, msx htheef fhmee fthevef ha avsxvsu qhzzhs uehbsx jvth fhzehse.theee mee zmsl ieseavtf ha ftmlvsu bsvtex vs thbuh tvzef, mf vt heycf th amqe qhmyyesuvsu fvtbmtvhsf jvth qhbemue. the vzchetmsqe ha ftmlvsu bsvtex hmf ftebqp m qhhex mzhsu zmsl cehcye thehbuhhbt hvfthel.peecvsu zezhevef ha jhmt je hmoe mqqhzcyvfhex thehbuhhbt hvfthel qms heyc bf fee hhj vsxvovxbmyf msx qhzzbsvtvef hmoe ceefeoeeex thehbuh thbuh tvzef msx vsth m ievuhtee abtbee

Since t,h,e are mapped already, now the word thmt can be nothing but “that”, so -

m=a

and the text –

the bsvtl vf fteesuth effal ekcyavsf that the chjee ha aevesxfhvc heycf bf th ie ftehsu aehz haexfhvcf. aevesxf qas uvoe bf the fteesuth je seex jhes yvae uetf thbuh. vs axxvtvhs, cehcye ekceeff bsvtl thehbuh yhoe,aazvyl, aevesxf, asx htheef fhaee fthevef ha avsxvsu qhzzhs uehbsx jvth fhzehse.theee aee zasl ieseavtf ha ftalvsu bsvtex vs thbuh tvzef, af vt heycf th aaqe qhayyesuvsu fvtbatvhsf jvth qhbeaue. the vzchetasqe ha ftalvsu bsvtex haf ftebqp a qhhex azhsu

zasl cehcye thehbuhhbt hvfthel.peecvsu zezhevef ha jhat je haoe aqqhzcyvfhex thehbuhhbt hvfthel qas heyc bf fee hhj vsxvovxbayf asx qhzzbsvtvef haoe ceefeoeeex thehbuh thbuh tvzef asx vsth a ievuhtee abtbee

Now “jhat” should be equal to “what”, so -

j = w

and the text –

the bsvtl vf fteesuth effal ekcyavsf that the chwee ha aevesxfhvc heycf bf th ie ftehsu aehz haexfhvcf. aevesxf qas uvoe bf the fteesuth we seex whes yvae uetf thbuh. vs axxvtvhs, cehcye ekceeff bsvtl thehbuh yhoe,aazvyl, aevesxf, asx htheef fhaee fthevef ha avsxvsu qhzzhs uehbsx wvth fhzehse.theee aee zasl ieseavtf ha ftalvsu bsvtex vs thbuh tvzef, af vt heycf th aaqe qhayyesuvsu fvtbatvhsf wvth qhbeaue. the vzchetasqe ha ftalvsu bsvtex haf ftebqp a qhhex azhsu zasl cehcye thehbuhhbt hvfthel.peecvsu zezhevef ha what we haoe aqqhzcyvfhex thehbuhhbt hvfthel qas heyc bf fee hhw vsxvovxbayf asx qhzzbsvtvef haoe ceefeoeeex thehbuh thbuh tvzef asx vsth a ievuhtee abtbee

Now “haoe” should be equal to “have”, so -

o = v

and the text –

the bsvtl vf fteesuth effal ekcyavsf that the chwee ha aevesxfhvc heycf bf th ie ftehsu aehz haexfhvcf. aevesxf qas uvve bf the fteesuth we seex whes yvae uetf thbuh. vs axxvtvhs, cehcye ekceeff bsvtl thehbuh yhve,aazvyl, aevesxf, asx htheef fhaee fthevef ha avsxvsu qhzzhs uehbsx wvth fhzehse.theee aee zasl ieseavtf ha ftalvsu bsvtex vs thbuh tvzef, af vt heycf th aaqe qhayyesuvsu fvtbatvhsf wvth qhbeaue. the vzchetasqe ha ftalvsu bsvtex haf ftebqp a qhhex azhsu zasl cehcye thehbuhhbt hvfthel.peecvsu zezhevef ha what we have aqqhzcyvfhex thehbuhhbt hvfthel qas heyc bf fee hhw vsxvvvxbayf asx qhzzbsvtvef have ceefeveeex thehbuh thbuh tvzef asx vsth a ievuhtee abtbee

In this way, we can get the rest of the mappings using relevant intuition. So, overall mapping –

'd': 'e',

'e': 'r',

'g': 't',

't': 'h',

'h': 'o',

'o': 'v',

'm': 'a',

'a': 'f',

'b': 'u’,

'u': 'g',

'i': 'b',

'v': 'i',

'f': 's',

's': 'n',

'c': 'p',

'x': 'd',

'l': 'y',

'j': 'w',

'y': 'l',

'k': 'x',

'z': 'm',

'q': 'c',

'p': 'k'

The ultimate solution is –

the unity is strength essay explains that the power of friendship helps us to be strong from hardships. friends can give us the strength we need when life gets tough. in addition, people express unity through love,family, friends, and others share stories of finding common

ground with someone.there are many benefits of staying united in tough times, as it helps to face challenging situations with courage. the importance of staying united has struck a chord among many people throughout history.keeping memories of what we have accomplished throughout history can help us see how individuals and communities have persevered through tough times and into a brighter future.

Code for this -

```
encryptedtext = '''gtd bsvgl vf fgedsugt dffml dkcymvsf gtmg gtd chjde ha
aevdsxftvc tdycf

bf gh id fgehsu aehz tmexftvcf. aevdsxf qms uvod bf gtd fgedsugt jd sddx

jtds yvad udgf ghbut. vs mxxvgvhs, cdhcyd dkcedff bsvgl gtehbut

yhod,amzvyl, aevdsxf, msx hgtdef ftmed fghevdf ha avsxvsu qhzzhs

uehbsx jvgt fhzdhsd.gtded med zmsl idsdavgf ha fgmlvsu bsvgdx vs

ghbut gvzdf, mf vg tdycf gh amqd qtmyydsuvsu fvgbmgvhsf jvgt

qhbemud. gtd vzchegmsqd ha fgmlvsu bsvgdx tmf fgebqp m qthex mzhsu

zmsl cdhcyd gtehbuthbg tvfghel.pddcvsu zdzhevdf ha jtmg jd tmod

mqqhzcyvftdx gtehbuthbg tvfghel qms tdyc bf fdd thj vsxvovxbmyf msx

qhzzbsvgvdf tmod cdefdodedx gtehbut ghbut gvzdf msx vsgh m ievutgde

abgbed.'''


mappings = {

    'd': 'e', 'e': 'r', 'g': 't', 't': 'h', 'h': 'o', 'o': 'v', 'm': 'a', 'a': 'f',

    'b': 'u', 'u': 'g', 'i': 'b', 'v': 'i', 'f': 's', 's': 'n', 'c': 'p', 'x': 'd',

    'l': 'y', 'j': 'w', 'y': 'l', 'k': 'x', 'z': 'm', 'q': 'c', 'p': 'k'
```

```
}

decryptedtext = sub_cipher(encryptedtext, mappings)

print("Decrypted Text : ", decryptedtext)



freq_count= frequency(decryptedtext)

print("frequency list : ")



for letter, freq in sorted(freq_count.items()):

    print(f"{letter}: {freq}")
```

b.

Given encrypted message is -

exupziu kxwqxagxom, upm gxsm zs l amtwzo exgg rmqzfm kigg

lok xolquxjm.lgwz, l kxwqxagxomk amtwzo qlo qzoutzg lok plokgm

upm wxuiluxzo zs gxjxoh xo l wzapxwuxqlumk elc uplo upzwm epz

kz ozu.fztmzjmt, xs czi pljm l aglo lok czi elou

uz xfagmfmou xu xo czit gxsm upmo czi ommk kxwqxagxom.

xu flnmw upxohw mlwc szt czi uz plokgm lok iguxflumgc

rtxoh wiqqmww uz czit gxsm.xs ulgn lrziu upm ucamw zs

kxwqxagxom, upmo upmc ltm hmomtlggc zs uez ucamw. sxtwu zom

xw xokiqmk kxwqxagxom lok upm wmqzok zom xw wmgs-

kxwqxagxom.xokiqmk kxwqxagxom

xw wzfmupxoh uplu zupmtw ulihpu iw zt em gmlto rc

wmmxoh zupmtw. epxgm wmgs-kxwqxagxom qzfmw stzf exupxo lok

em gmlto xu zo zit zeo wmgs. wmgs-kxwqxagxom tmyixtmw l gzu

zs fzuxjluxzo lok wiaaztu stzf zupmtw.lrzjm lgg, szggzexoh czit klxgc

wqpmkigm exupziu loc fxwulnm xw lgwz altu zs rmxoh kxwqxagxomk.

Here “upm” appears numerous times which can be nothing but “the”, so-

u=t

p=h

m=e

So now replacing with these mapping, the text becomes –

exthzit kxwqxagxoe, the gxse zs l aetwzo exgg reqzfe kigg

lok xolqtxje.lgwz, l kxwqxagxoek aetwzo qlo qzottzg lok hlokge

the wxtiltxzo zs gxjxoh xo l wzahxwtxqltek elc thlo thzwe ehz

kz ozt.fztezjet, xs czi hlje l aglo lok czi elot

tz xfagefeot xt xo czit gxse theo czi oek kxwqxagxoe.

xt flnew thxohw elwc szt czi tz hlokge lok igtxfltegc

rtxoh wiqqeww tz czit gxse.xs tlgn lrzit the tcaew zs

kxwqxagxoe, theo thec lte heoetlggc zs tez tcaew. sxtwt zoe

xw xokiqek kxwqxagxoe lok the weqzok zoe xw wegs-

kxwqxagxoe.xokiqek kxwqxagxoe

xw wzfethxoh thlt zthetw tlihht iw zt ee gelto rc

wexoh zthetw. ehxge wegs-kxwqxagxoe qzfew stzf exthxo lok

ee gelto xt zo zit zeo wegs. wegs-kxwqxagxoe teyixtew l gzt

zs fztxjltxzo lok wiaaztt stzf zthetw.lrzje lgg, szggzexoh czit klxgc

wqhekige exthzit loc fxwtlne xw lgwz altt zs rexoh kxwqxagxoek.

Since t,h,e are mapped already, now the word “upmc” can be nothing but “they”, so -

C=y

and the text –

exthzit kxwqxagxoe, the gxse zs l aetwzo exgg reqzfe kigg

lok xolqtxje.lgwz, l kxwqxagxoek aetwzo qlo qzottzg lok hlokge

the wxtiltxzo zs gxjxoh xo l wzahxwtxqltek ely thlo thzwe ehz

kz ozt.fztezjet, xs yzi hlje l aglo lok yzi elot

tz xfagefeot xt xo yzit gxse theo yzi oek kxwqxagxoe.

xt flnew thxohw elwy szt yzi tz hlokge lok igtxfltegy

rtxoh wiqqeww tz yzit gxse.xs tlgn lrzit the tyaew zs

kxwqxagxoe, theo they lte heoetlggy zs tez tyaew. sxtwt zoe

xw xokiqek kxwqxagxoe lok the weqzok zoe xw wegs-

kxwqxagxoe.xokiqek kxwqxagxoe

xw wzfethxoh thlt zthetw tlihht iw zt ee gelto ry

wexoh zthetw. ehxge wegs-kxwqxagxoe qzfew stzf exthxo lok

ee gelto xt zo zit zeo wegs. wegs-kxwqxagxoe teyixtew l gzt

zs fztxjltxzo lok wiaaztt stzf zthetw.lrzje lgg, szggzexoh yzit klxgy

wqhekige exthzit loy fxwtlne xw lgwz altt zs rexoh kxwqxagxoek.

In this way, we can get the rest of the mappings using relevant intuition. So, overall mapping –

'm':'e',

'l':'a',

'o':'n',

'k':'d',

'u':'t',

'p':'h',

'm':'e',

'g':'l',

'j':'v',

't':'r',

'r':'b',

'z':'o',

'w':'s',

'a':'p',

'q':'c',

'x':'i',

'i':'u',

'e':'w',

's':'f',

'f':'m',

'h':'g',

'c':'y',

'n':'k',

'y':'q'

The ultimate solution is –

without discipline, the life of a person will become dull and inactive.also, a disciplined person can control and handle the situation of living in a sophisticated way than those who do not.moreover, if you have a plan and you want to implement it in your life then you need discipline. it makes things easy for you to handle and ultimately bring success to your life.if talk about the types of discipline, then they are generally of two types. first one is induced discipline and the second one is self-discipline.induced discipline is something that others taught us or we learn by seeing others. while self-discipline comes from within and we learn it on our own self. self-discipline requires a lot of motivation and support from others.above all, following your daily schedule without any mistake is also part of being disciplined.

Code for this -

```
encryptedtext = '''exupziu kxwqxagxom, upm gxsm zs l amtwzo exgg rmqzfm kigg

lok xolquxjm.lgwz, l kxwqxagxomk amtwzo qlo qzoutzg lok plokgm

upm wxuiluxzo zs gxjxoh xo l wzapxwuxqlumk elc uplo upzwm epz

kz ozu.fztmzjmt, xs czi pljm l aglo lok czi elou

uz xfagmfmou xu xo czit gxsm upmo czi ommk kxwqxagxom.

xu flnmw upxohw mlwc szt czi uz plokgm lok iguxflumgc

rtxoh wiqqmww uz czit gxsm.xs ulgn lrziu upm ucamw zs

kxwqxagxom, upmo upmc ltm hmomtlggc zs uez ucamw. sxtwu zom
```

```
xw xokiqmk kxwqxagxom lok upm wmqzok zom xw wmgs-

kxwqxagxom.xokiqmk kxwqxagxom



xw wzfmupxoh uplu zupmtw ulihpu iw zt em gmlto rc

wmmxoh zupmtw. epxgm wmgs-kxwqxagxom qzfmw stzf exupxo lok

em gmlto xu zo zit zeo wmgs. wmgs-kxwqxagxom tmyixtmw l gzu

zs fzuxjluxzo lok wiaaztu stzf zupmtw.lrzjm lgg, szggzexoh czit klxgc

wqpmkigm exupziu loc fxwulnm xw lgwz altu zs rmxoh kxwqxagxomk.'''



mappings = {

    'm':'e',

    'l':'a',

    'o':'n',

    'k':'d',

    'u':'t',

    'p':'h',

    'm':'e',

    'g':'l',

    'j':'v',

    't':'r',

    'r':'b',

    'z':'o',

    'w':'s',

    'a':'p',

    'q':'c',

    'x':'i',

    'i':'u',
```

```
    'e':'w',
    's':'f',
    'f':'m',
    'h':'g',
    'c':'y',
    'n':'k',
    'y':'q'
}
print()

decryptedtext = sub_cipher(encryptedtext, mappings)
print("Decrypted Text : ", decryptedtext)

freq_count= frequency(decryptedtext)
print("frequency list : ")

for letter, freq in sorted(freq_count.items()):
    print(f"{letter}: {freq}")
```

C.

Given encrypted message is –

AUHC MVKFC V BYZUGC V IZMC CJ GUMBZYAZD UKUVM.

VC HZZGZB CJ GZ, V HCJJB PD CFZ VYJM KUCZ AZUBVMK CJ CFZ

BYVWZ UMB OJY U IFVAZ, V TJNAB MJC ZMCZY.

OJY CFZ IUD, VC IUH PUYYZB CJ GZ.

Here “V” appears numerous times which can be nothing but “I”, so-

V=I

So now replacing with these mapping, the text becomes –

AUHC MIKFC I BYZUGC I IZMC CJ GUMBZYAZD UKUIM.

IC HZZGZB CJ GZ, I HCJJB PD CFZ IYJM KUCZ AZUBIMK CJ CFZ

BYIWZ UMB OJY U IFIAZ, I TJNAB MJC ZMCZY.

OJY CFZ IUD, IC IUH PUYYZB CJ GZ

In this way, we can get the rest of the mappings using relevant intuition. So, overall mapping –

'V':'I',

'C':'T',

'Z' :'E',

'F':'H',

'J':'O',

'G':'M',

'D':'Y',

'M':'N',

'Y':'R',

'K':'G',

'U':'A',

'B':'D',

'I':'W',

'H':'S',

'A':'I',

'O':'F',

'P':'B',

'W':'V',

'T':'C',

'N':'U',

'E':'X',

'I':'P',

'Q':'J',

'S':'Q',

'R':'Z',

'X':'K'

The ultimate solution is –

LAST NIGHT I DREAMT I WENT TO MANDERLEY AGAIN. IT SEEMED TO ME, I STOOD BY THE IRON

GATE LEADING TO THE DRIVE AND FOR A WHILE, I COULD NOT ENTER. FOR THE WAY, IT WAS BARRED TO ME.

Code for this -

```
encryptedtext = '''AUHC MVKFC V BYZUGC V IZMC CJ GUMBZYAZD UKUVM.

VC HZZGZB CJ GZ, V HCJJB PD CFZ VYJM KUCZ AZUBVMK CJ CFZ

BYVWZ UMB OJY U IFVAZ, V TJNAB MJC ZMCZY.

OJY CFZ IUD, VC IUH PUYYZB CJ GZ.'''

# LDTYXHMSWOGPNUFBJZQCAIVKRE

mappings = {

   'v':'i',

    'c':'t',

    'z' :'e',

    'f':'h',

    'j':'o',

    'g':'m',

    'd':'y',

    'm':'n',

    'y':'r',

    'k':'g',

    'u':'a',

    'b':'d',

    'i':'w',

    'h':'s',

    'a':'l',
```

```
    'o':'f',
    'p':'b',
    'w':'v',
    't':'c',
    'n':'u',
    'e':'x',
    'l':'p',
    'q':'j',
    's':'q',
    'r':'z',
    'x':'k'
}
print()

decryptedtext = sub_cipher(encryptedtext,mappings)
print("Decrypted Text : ", decryptedtext)

freq_count= frequency(decryptedtext)
print("frequency list : ")

for letter, freq in sorted(freq_count.items()):
    print(f"{letter}: {freq}")
```

d.

Given encrypted message is –

JGRMQOYGHMVBJ WRWQFPW HGF FDQGFPFZR KBEEBJIZQ QO CIBZK. LFAFGQVFZFWW, EOG WOPF GFHWOL PHLR LOLFDMFGQW BLWBWQ OL KFWBYLBLY LFS FLJGRMQBOL WJVFPFW QVHQ WFFP QO QVFP QO CF POGF WFJIGF QVHL HLR OQVFG WJVFPF OL FHGQV. QVF ILEOGQILHQF QGIQV VOSFAFG BW QVHQ WIJV WJVFPFW HGF IWIHZZR QGBABHZ QO CGFHX.

mapping –

A:V

B:I

C:B

D:X

E:F

F:E

G:R

H:A

I:U

J:C

K:D

L:N

M:P

N:Z

O:O

P:M

Q:T

R:Y

S:W

T:Q

U:J

V:H

W:S

X:K

Y:G

Z:L

The ultimate solution is –

CRYPTOGRAPHIC SYSTEMS ARE EXTREMELY DIFFICULT TO BUILD. NEVERTHELESS, FOR SOME REASON MANY NONEXPERTS INSIST ON

DESIGNING NEW ENCRYPTION SCHEMES THAT SEEM TO THEM TO BE MORE SECURE THAN ANY OTHER SCHEME ON EARTH. THE UNFORTUNATE TRUTH HOWEVER IS THAT SUCH SCHEMES ARE USUALLY TRIVIAL TO BREAK.

Code for this -

```
encryptedtext = '''JGRMQOYGHMVBJ WRWQFPW HGF FDQGFPFZR KBEEBJIZQ QO CIBZK.

LFAFGQVFZFWW, EOG WOPF GFHWOL PHLR LOLFDMFGQW BLWBWQ OL

KFWBYLBLY LFS FLJGRMQBOL WJVFPFW QVHQ WFFP QO QVFP QO CF

POGF WFJIGF QVHL HLR OQVFG WJVFPF OL FHGQV. QVF ILEOGQILHQF

QGIQV VOSFAFG BW QVHQ WIJV WJVFPFW HGF IWIHZZR QGBABHZ QO

CGFHX.'''



mappings = {

   'f':'e',

    'q':'t',

    'o':'o',

    'v':'h',

    'h':'a',

    'g':'r',

    'd':'x',

    'p':'m',

    'z':'l',

    'r':'y',

    'l':'n',

    'e':'f',

    'w':'s',
```

```
    'm':'p',
    'a':'v',
    'j':'c',
    'b':'i',
    'c':'b',
    'i':'u',
    'y':'g',
    'k':'d',
    's':'w',
    'x':'k'
}
print(len(mappings))

decryptedtext = sub_cipher(encryptedtext, mappings)
print("Decrypted Text : ", decryptedtext)

freq_count= frequency(decryptedtext)
print("frequency list : ")

for letter, freq in sorted(freq_count.items()):
    print(f"{letter}: {freq}")
```

TASK 3

Following is the code for the Vignere cryptosystem which implements the encryption as well as decryption method. The program asks for a key and a plaintext initially and then a choice for encryption or decryption is placed before the user depending upon which the program proceeds and performs the expected operation.

```
#include <bits/stdc++.h>
using namespace std;

string vignere_encrypt(string plainText, string key)
{
    string encryptedText;

    int newkey[plainText.size()];
    for (int i = 0, j = 0; i < plainText.size(); i++, j++)
    {
        if (plainText[i] == ' ')
        {
            newkey[i] = -1;
            j--;
        }
        else
            newkey[i] = key[j % key.size()] - 'A';
    }
    // for (int i=0;plaineText.size();i++)
    //     cout << int(plainText[i]) << " ";
    // cout <<endl;
    // for (int i=0;plaineText.size();i++)
    //     cout << newkey[i] << " ";
    // cout <<endl;
```

```
    for (int i = 0; i < plainText.size(); i++)
    {
        if (newkey[i] == -1)
            encryptedText += ' ';
        else
            encryptedText += (int(plainText[i] - 65) + newkey[i]) % 26 + 'A';
    }
    return encryptedText;
}

string vignere_decrypt(string plainText, string key)
{
    string decryptedText;

    int newkey[plainText.size()];
    for (int i = 0, j = 0; i < plainText.size(); i++, j++)
    {
        if (plainText[i] == ' ')
        {
            newkey[i] = -1;
            j--;
        }
        else
            newkey[i] = key[j % key.size()] - 'A';
    }
    for (int i = 0; i < plainText.size(); i++)
    {
        if (newkey[i] == -1)
            decryptedText += ' ';
        else
            decryptedText += (int(plainText[i] - 65) - newkey[i] + 26) % 26 + 'A';
    }
    return decryptedText;
}

int main()
{

        string plainText, key;
        cout << "Enter key :    ";
        getline(cin, key);
        int choice;
        cout << "Press 0 for Encrypt,1 for Decrypt :    ";
        cin >> choice;
        cin.ignore();
        if (!choice)
        {
            cout << "Enter text :   ";
            getline(cin, plainText,'\n');
            cout << "Encrypted message :    " << vignere_encrypt(plainText, key);
        }
```

```
        else
        {
            cout << "Enter text :   ";
            getline(cin, plainText,'\n');


            cout << "Decrypted message :    " << vignere_decrypt(plainText, key);
        }


    return 0;
}

//key : SUSTCSE

//for encryption
// input : CSE FINAL YEAR THEORY COURSE INTRODUCTION TO COMPUER SECURITY AND
FORENSICS
// output :UMW YKFED SWTT LLWIJR EGYJMW BPLVGXMVVASF NG VQETMYJ LGUYJCLR CFH
XIJXPKMUM

//for decryption
// input : UMW YKFED SWTT LLWIJR EGYJMW BPLVGXMVVASF NG VQETMYJ LGUYJCLR CFH
XIJXPKMUM
// output : CSE FINAL YEAR THEORY COURSE INTRODUCTION TO COMPUER SECURITY AND
FORENSICS
```

TASK 4

Following is the code for the Hill Cipher cryptosystem which implements the encryption method. The program asks for a key and a plaintext initially and then provides the encrypted message as output -

```
#include <iostream>
#include <vector>
#include <cmath>

using namespace std;

vector<vector<int>> key_matrix(const string& key, int size) {
    vector<vector<int>> K;
    vector<int> tmp;
    int cnt = 0;

    for (char k : key) {
        if (k != ' ') {
            tmp.push_back(k - 'A');
            cnt += 1;

            if (cnt == size) {
                K.push_back(tmp);
                tmp.clear();
                cnt = 0;
            }
        }
    }

    return K;
}
```

```cpp
string convert(const vector<int>& v, const string& text) {
    string result = "";
    int index = 0;

    for (char ch : text) {
        char nxt = ch;
        if (nxt != ' ') {
            nxt = static_cast<char>(round(v[index])) + 'A';
            index += 1;
        }
        result += nxt;
    }

    return result;
}

string encrypt(const string& key, const string& plaintext) {
    vector<int> x;
    for (char ch : plaintext) {
        if (ch != ' ') {
            x.push_back(ch - 'A');
        }
    }

    int size = x.size();
    vector<vector<int>> K = key_matrix(key, size);

    vector<int> c(size, 0);
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            c[i] += K[i][j] * x[j];
        }
        c[i] %= 26;
    }

    string ciphertext = convert(c, plaintext);

    return ciphertext;
}

int main() {
    string key,plaintext;
    cout << "Enter key :    ";
    getline(cin, key);
    cout << "Enter plaintext :   ";
    getline(cin, plaintext);
    string encrypted_text = encrypt(key, plaintext);

    cout << "Encrpyted Text :   "<<encrypted_text << endl;
```

```
    return 0;
}
//key : AWESOME INTRODUCTION TO COMPUTER SECURITY AND FORENSICS
//input : SUST CSE
//output : WJCT KZU
```