PROGRAMMING ASSIGNMENT

CRYPTOGRAPHY (BITS F463), SEMESTER-I, 20.09.2017

This Assignment is submitted for the partial fulfillment of the Cryptography Course(BITS 463)

Instructor- Dr. Ashutosh Bhatia

Submitted By:

ANJAN DAS
(2015A7PS0150P)

TABLE OF CONTENTS

ENCRYPTED TEXT 1

ENCRYPTED TEXT WITHOUT SPECIAL CHARACTERS 2

DECRYPTED PLAIN TEXT 2

DECRYPTED TEXT WITH SPECIAL CHARACTERS 3

GIVEN 3

METHODOLOGY 4

- STEP 1 4
- STEP 2 5
- STEP 3 8
- STEP 4 18

ENCRYPTED TEXT

“czuyw u dipniye phgdcaocltr pckp uamlnf hh rv htltmvmyu arz oq tbicta gnrzuta zccrtt fsr hz yczgn waazoror? ioflq t frvtaare hlceo zgcsiax azdr zhp aciyrzntcp os nhumeytlnqbhx arttpnpxm rvq ioxiaz og evzh tdrtm npvre tn gkokp okiyg nl xvdbod zf gailouzs lnq tm vuczy tnfbxv if g ntnrmyvvgn cpngnlp iqjiyg ztwyqak oc a gpyebvkts crgnlzl cocd ckitmfyoc? hbp gzouz wp dvlnzvtaidh oxnnmrt a reancemye cznfvcfcf gno iamyctvmeyt zbhu iaj bft n vfvdrxlj cbgmkzhitpd onn ywyroh lngalitk udiaz zrknje? lrr nhumeytlnqbhx iaj rpafhhzvt onnoziukqore higa grbrxillvlnzk, zkcsaabmkqp bipw by fzdvtg mevgaj? kbalo a ztwyqak egee uy jivj tz hnoy diqk ies bph umpostoal? wfcyj a xapacem ugvp brecvnf. ioflq t grkuonp mndy dqfzavef? viltq g mlcubhv jrripvr bn diqk ies bph umpostoal? wfcyj a xapacem rxrznrhojtl lrpe jbfc bb otdeyy? wfcyj a xapacem pump uc pckp vjels gauk pnbe yog uyzvt vrzgetgdmq oneo vm ce iqbaycr. viltq irpagbpvtl kmprtx ziwz g spt by zzfrj rflrl? uim jk egea mbv ubyt nrrtnzdr gmznt nm scg, vadsvoy jtnbed purmzkf zhlt thpvza uuc nrnlfvf?”

ENCRYPTED TEXT WITHOUT SPECIAL CHARACTERS

“czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrhzyczgnwaazororioflqtfrvtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqioxiazogevzhtdrtmnpvretngkokpokiygnlxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiqjiygztwyqakocagpyebvktscrgnlzlcocdckitmfyochbpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvcfcfgnoiamyctvmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnywyrohlngalitkudiazzrknjelrrnhumeytlnqbhxiajrpafhhzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfzdvtgmevgajkbaloaztwyqakegeeuyjivjtzhnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpmndydqfzavefviltqgmlcubhvjrripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjaxapacempumpucpckpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgsptbyzzfrjrflrluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfvf”

DECRYPTED PLAIN TEXT

“couldamachinecommunicatewithhumansonanunlimitedsetoftopicsthroughfluentuseofhumanlanguagecouldalanguageusingmachinegivetheappearanceofunderstandingsentencesandcomingupwithideaswhileintruthbeingasdevoidofthoughtsandasemptyinsideasanin eteenthcenturyaddingmachineoratwentiethcenturywordprocessorhowmightwedistinguishbetweenagenuinelyconsciousandintelligentmindandbutacleverlyconstructedbuthollowlanguageusingfacadeareunderstandingandreasoningincompatiblewithmaterialisticmechanisticviewoflivingbeingscouldamachineeverbesaidtohavemadeitsowndecisionscouldamachinehavebeliefscouldamachinemakemistakescouldamachinebelieveitmadeitsowndecisionscouldamachineerroneouslyfreewilltoitselfcouldamachinecomeupwithideasthathavenotbeingprogrammedintoitinadvancecouldcreativelyemergefromasetoffixedrulesareweeventhemostcreativeamongusbutpassiveslavesphysicsthatgovernourneurons”

DECRYPTED TEXT WITH SPECIAL CHARACTERS

“Could a machine communicate with humans on an unlimited set of topics through fluent use of human language? Could a language using machine give the appearance of understanding sentences and coming up with ideas while in truth being as devoid of thought and as empty inside as a nineteenth century adding machine or a twentieth century word processor? How might we distinguish between a genuinely conscious and intelligent mind and but a cleverly constructed but hollow language using facade are understanding and reasoning incompatible with materialistic mechanistic view of living beings could a machine never be said to have made its own decisions could a machine have beliefs could a machine make mistakes could a machine believe it made its own decisions could a machine erroneously free will to itself could a machine come up with ideas that have not being programmed into it in advance could creatively emerge from a set of fixed rules are we even the most creative among us but passive slaves physics that govern our neurons.”

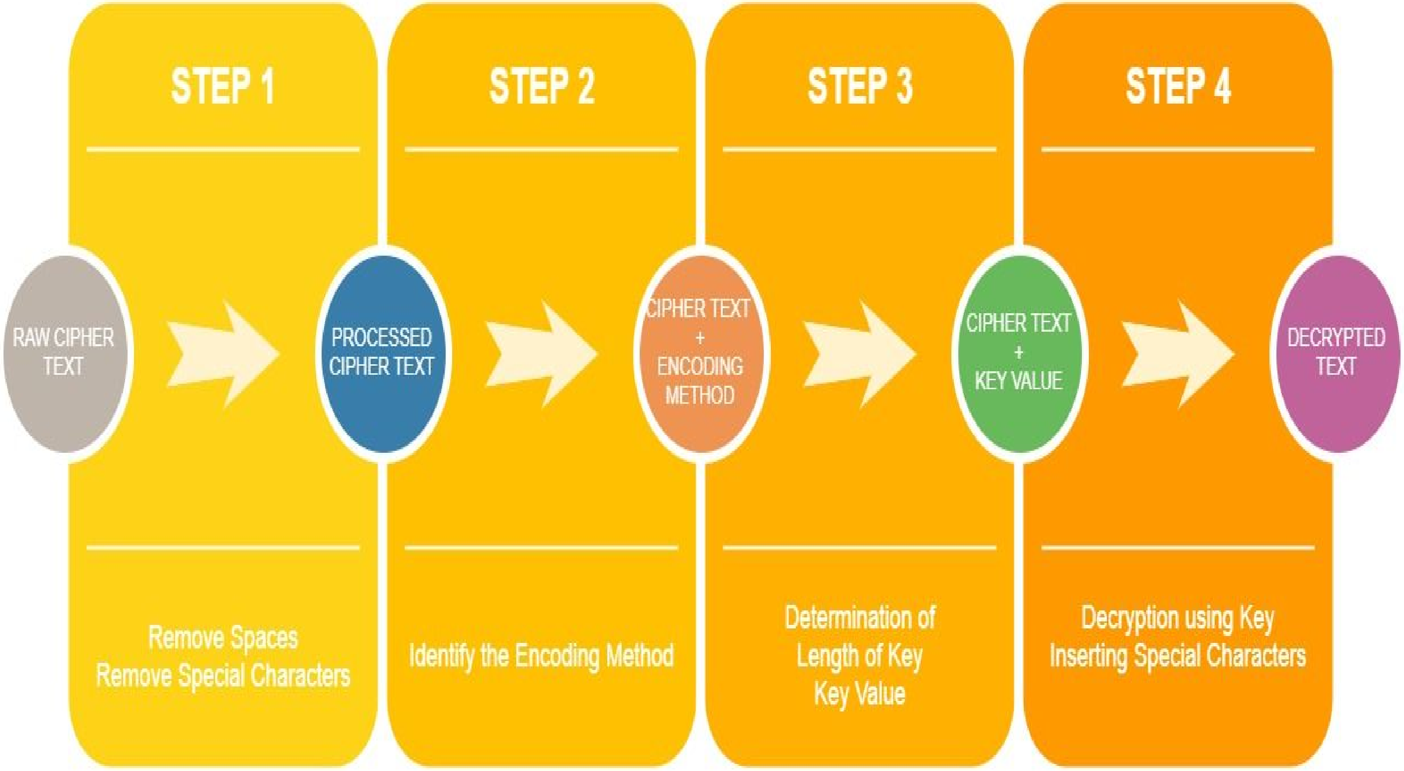
GIVEN

It is already given that the encryption is done using either of the following methods:

1. Shift Cipher
2. Substitution Cipher
3. Vigenere Cipher
4. Transposition Cipher

It is also given that special characters (characters other then a-z) are mapped to the same value. And there is no differentiation between “small” and “capital” (a = 0 …… z = 25)

METHODOLOGY


STEP 1
STEP 2
STEP 3
STEP 4
RAW CIPHER TEXT
PROCESSED CIPHER TEXT
CIPHER TEXT + ENCODING METHOD
CIPHER TEXT + KEY VALUE
DECRYPTED TEXT
Remove Spaces
Remove Special Characters
Identify the Encoding Method
Determination of Length of Key
Key Value
Decryption using Key
Inserting Special Characters

STEP 1

The first and foremost step is to remove the Special Characters form the Cipher Text in order to make it easier for us to process.

C++ Code to remove the spaces and special characters:

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

// Function to remove all spaces from a given string
void removeSpecials(char *str)
{
    // To keep track of non-space character count
    int count = 0;

    // Traverse the given string. If current character
    // is not space, then place it at index 'count++'
    for (int i = 0; str[i]; i++)
```

```
        if (str[i] >= 'a' && str[i] <= 'z')
            str[count++] = str[i]; // here count is
    str[count] = '\0';
}

// Driver program to test above function
int main()
{
    char str[] = "czuyw u dipniye phgdcaocltr pckp uamlnf hh rv htltmvmyu arz oq tbicta gnrzuta zccrtt fsr hz yczgn waazoror?ioflq t frvtaare hlceo zgcsiax azdr zhp aciyrzntcp os nhumeytlnqbhx arttpnpxm rvq ioxiaz og evzh tdrtm npvretn gkokp okiyg nl xvdbod zf gailouzs lnq tm vuczy tnfbxv if g ntnrmyvvgn cpngnlp iqjiyg ztwyqak oc agpyebvkts crgnlzl cocd ckitmfyoc? hbp gzouz wp dvlnzvtaidh oxnnmrt a reancemye cznfvcfcf gno iamyctvmeytzbhu iaj bft n vfvdrxlj cbgmkzhitpd onn ywyroh lngalitk udiaz zrknje? lrr nhumeytlnqbhx iaj rpafhhzvtonnoziukqore higa grbrxillvlnzk, zkcsaabmkqp bipw by fzdvtg mevgaj? kbalo a ztwyqak egee uy jivj tz hnoydiqk ies bph umpostoal? wfcyj a xapacem ugvp brecvnf. ioflq t grkuonp mndy dqfzavef? viltq g mlcubhv jrripvrbn diqk ies bph umpostoal? wfcyj a xapacem rxrznrhojtl lrpe jbfc bb otdeyy? wfcyj a xapacem pump uc pckpvjels gauk pnbe yog uyzvt vrzgetgdmq oneo vm ce iqbaycr. viltq irpagbpvtl kmprtx ziwz g spt by zzfrj rflrl? uimjk egea mbv ubyt nrrtnzdr gmznt nm scg, vadsvoy jtnbed purmzkf zhlt thpvza uuc nrnlfvf?";
    removeSpecials(str);
    cout << str;
    return 0;
}
```

Output: Run on IDE

```
•   czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrhzyczgnwaazor
    orioflqtfrvtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqioxiazogevzhtdr
    tmnpvretngkokpokiygnlxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiqjiygztwyqako
    cagpyebvktscrgnlzlcocdckitmfyochbpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvcfcfgnoiamyct
    vmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnywyrohlngalitkudiazzrknjelrrnhumeytlnqbhxiajrpafh
    hzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfzdvtgmevgajkbaloaztwyqakegeeuyjivjtz
    hnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpmndydqfzavefviltqgmlcubhvjr
    ripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjaxapacempumpucpc
    kpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgsptbyzzfrjrfl
    rluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfvf
```

So, this is the text which we need to decrypt first and then we can go on to insert the special characters back at the corresponding index to get the complete Decrypted Text.

STEP 2

The next step is to identify the encoding method which has been used in making the Cipher text.

To rule out Transposition Cipher, we take advantage of the fact that, English text has a very specific frequency distribution that is not changed by transposition ciphers. All other ciphers change this distribution, so the frequencies can be used to differentiate them. If the frequency distribution looks exactly like a piece of english text but it is still unreadable we can conclude it is probably a transposition cipher, otherwise we move onto the next step.

Fig.1. And Fig.2. show the difference between the Monogram Frequency of the given Cipher text and general English Language. Hence, we can infer that Transposition Cipher hasn’t been used since, the the Monogram Frequency pattern has been broken.

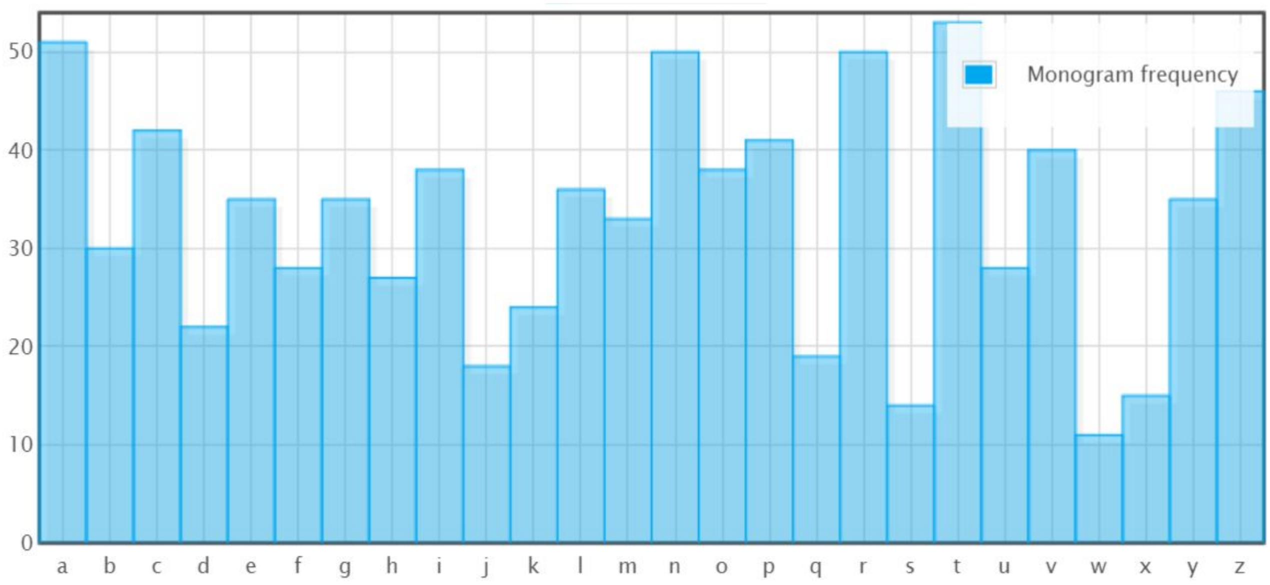


Fig.1. Monogram Frequency of the given Cipher Text

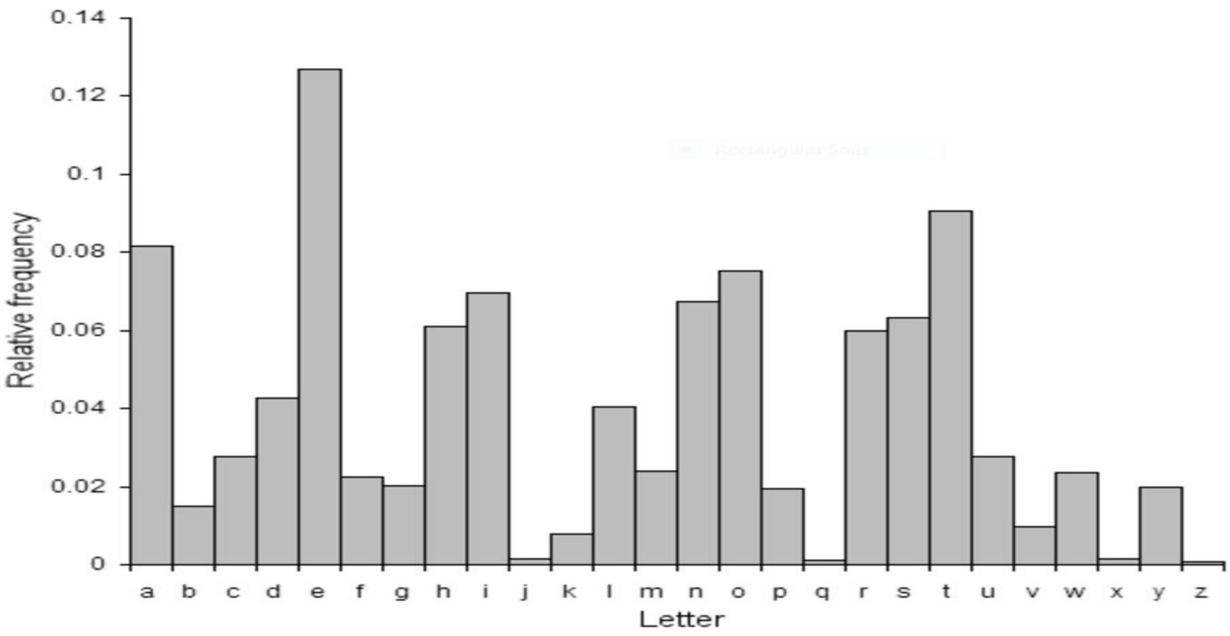


Fig.2. Monogram Frequency of general English Language(Source)

Now, we check if the method is Shift Cipher for which the following C++ code would suffice. What the code does is, it checks for all the keys from 1 to 25(which is used in Shift Cipher) to check if any meaningful message is formed. Just for the sake of Simplicity, I have only used the first few lines and check.

```
#include<iostream>

using namespace std;

int main()
{
    string message("czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqt");//using the initial words of thre cipher text to check for Caesar Cipher.
    char ch;
    int i, key;
    for(int j=0;j<26;j++)//loop to check for all the keys in the form of j
    {
    for(i = 0; message[i] != '\0'; ++i){
        ch = message[i];
        if(ch >= 'a' && ch <= 'z'){
            ch = ch - j;
            if(ch < 'a'){
                ch = ch + 'z' - 'a' + 1;
            }
            message[i] = ch;
        }
            message[i] = ch;
        }
    }
    cout << j << "\t"<< message<<endl;
    }
    return 0;
}
```

Output: Run on IDE

```
0   czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqt
1   bytxvtchomhxdogfcbznbksqobjotzlkmeggqugskslulxtzqynps
2   zwrvtrafmkfvbmedazxlziqomzhmrxjikceeoseqiqjsjvrxowlnq
3   wtosqoxcjhcsyjbaxwuiwfnljwejougfhzbblpbnfngpgsoultikn
4   spkomktyfdyoufxwtsqesbjhfsafkqcbdvxxhlxjbjclcokqhpegj
5   nkfjhfotaytjpasronlznwecanvaflxwyqsscgsewexgxjflckzbe
6   hezdbzinusndjumlihfthqywuhpuzfrqskmmwamyqyrardzfwetvy
7   axswusbgnlgwcnfebaymajrpnainsykjldffptfrjrktkwsypxmor
8   spkomktyfdyoufxwtsqesbjhfsafkqcbdvxxhlxjbjclcokqhpegj
9   jgbfdbkpwupflwonkjhvjsaywjrwbhtsumooycoasatctfbhygvxa
10  zwrvtrafmkfvbmedazxlziqomzhmrxjikceeoseqiqjsjvrxowlnq
11  olgkigpubzukqbtspomaoxfdbowbgmyxzrttdhtfxfyhykgmdlacf
12  czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqt
13  pmhljhqvcavlrcutqpnbpygecpxchnzyasuueiugygzizlhnembdg
14  bytxvtchomhxdogfcbznbksqobjotzlkmeggqugskslulxtzqynps
15  mjeigenszxsiozrqnmkymvdbzmuzekwvxprrbfrdvdwfwiekbjyad
16  wtosqoxcjhcsyjbaxwuiwfnljwejougfhzbblpbnfngpgsoultikn
17  fcxbzxglsqlbhskjgfdrfowusfnsxdpoqikkuykwowpypbxducrtw
18  nkfjhfotaytjpasronlznwecanvaflxwyqsscgsewexgxjflckzbe
19  urmqomvahfaqwhzyvusgudljhuchmsedfxzzjnzldleneqmsjrgil
20  axswusbgnlgwcnfebaymajrpnainsykjldffptfrjrktkwsypxmor
21  fcxbzxglsqlbhskjgfdrfowusfnsxdpoqikkuykwowpypbxducrtw
22  jgbfdbkpwupflwonkjhvjsaywjrwbhtsumooycoasatctfbhygvxa
23  mjeigenszxsiozrqnmkymvdbzmuzekwvxprrbfrdvdwfwiekbjyad
24  olgkigpubzukqbtspomaoxfdbowbgmyxzrttdhtfxfyhykgmdlacf
25  pmhljhqvcavlrcutqpnbpygecpxchnzyasuueiugygzizlhnembdg
```

Hence, nothing meaningful could be derived from any of the output lines.

Now, we need to check if its `Substitution Cipher. By referring to the slides, and after following the process and making the guesses manually, nothing could be derived from, except some meaningless deductions.

Hence, the conclusion is that, since all the other three Cipher Methods are not involved, we need to check if its Vigenere Cipher or not. Intuition says that it must be, but again we cannot rely on intuition and we need to verify it.

STEP 3

In this step we need to find out the Encryption Key, now that we know that it might strongly be a Vigenere Cipher Text. First, we have to find out the length of the Key. This is calculated using Index of Coincidence(I.O.C.) which is abrupt whenever we pick letters from the Cipher text from the indices which are spaced by the length of the key(just the same as what is given in the slides). The following JAVA code suffices all that is needed for implementing the above mentioned procedure.

```
public class IC {

//Class Constructor
    public IC(){

    }

 //returns the IC of the input string
    public double calculate(String s)
    {

        int i;
        int N = 0;
        double sum = 0.0;
        double total = 0.0;
        s = s.toUpperCase();

        //initialize array of values to count frequency of each letter
        int[] values = new int[26];
        for(i=0; i<26; i++){
            values[i] = 0;
        }

        //calculate frequency of each letter in s
        int ch;
        for(i=0; i<s.length(); i++){
            ch = s.charAt(i)-65;
            if(ch>=0 && ch<26){
                values[ch]++;
                N++;
                }
        }

        //calculate the sum of each frequency
        for(i=0; i<26; i++){
            ch = values[i];
            sum = sum + (ch * (ch-1));
            }

        //divide by N(N-1)
        total = sum/(N*(N-1));

        //return the result
        return total;

    }
    public String everyNth(String str, int n) {
    if (n < 1) {
        throw new IllegalArgumentException("n must be greater than 0");
    }
    StringBuilder result = new StringBuilder();
    // The index of the previous match
    for(int i=0;i<str.length();i+=n)
    {
        result.append(str.charAt(i));
    }
    return result.toString();
}
    public static void main(String[] args)
    {
      IC test = new IC();
      String str1;
      double Sum=0.0,Avg, Diff;

      String
testString="czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrh
zyczgnwaazororioflqtfrvtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqiox
iazogevzhtdrtmnpvretngkokpokiygnlxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiq
jiygztwyqakocagpyebvktscrgnlzlcocdckitmfyochbpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvc
fcfgnoiamyctvmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnywyrohlngalitkudiazzrknjelrrnhumeytln
qbhxiajrpafhhzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfzdvtgmevgajkbaloaztwyqak
egeeuyjivjtzhnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpmndydqfzavefvil
tqgmlcubhvjrripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjaxap
acempumpucpckpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgs
ptbyzzfrjrflrluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfv
f";
      for(int i=2;i<testString.length();i++)
      {
          Sum=0;
          for(int j=0;j<i;j++)
          {
              str1 = test.everyNth(testString.substring(j),i);
              Sum += test.calculate(str1);
          }
          Avg = Sum/i;
          Diff = Math.abs(Avg-0.065);
          System.out.println(i + " " + Avg + Diff);
          if(Diff<0.005)
          System.out.println("Yes");
      }
    }
}
```

Output Excerpt(the real being too long): Run on IDE

```
2   0.044040339867333994    0.020959660132666008
3   0.04193749482581318     0.02306250517418682
4   0.04429517771703029     0.020704822282969715
5   0.05527651778785769     0.00972348221214231
6   0.04514926257884005     0.019850737421159954
7   0.04242647687088093     0.022573523129119075
8   0.04543502680956393     0.01956497319043607
9   0.04283677574551034     0.02216322425448966
10  0.06499316005471956     6.839945280445736E-6
Yes
11  0.04100080970046447     0.023999190299535532
12  0.04471737834413891     0.020282621655861095
13  0.042467062214478976    0.022532937785521026
14  0.043863161340686456    0.021136838659313546
15  0.05552098637398093     0.009479013626019071
16  0.045236319410847714    0.019763680589152288
17  0.04154508862368476     0.023454911376315245
18  0.04764107308048104     0.017358926919518965
19  0.043196264706562196    0.021803735293437806
20  0.06392782864705723     0.001072171352942769
Yes
21  0.04142172190952679     0.023578278090473213
22  0.041976444608023564    0.02302355539197644
23  0.04231691645879289     0.022683083541207114
24  0.046047307812013695    0.018952692187986307
25  0.053308887191240155    0.011691112808759847
26  0.04484608528726175     0.020153914712738252
27  0.04100457984866587     0.02399542015133413
28  0.043090206049049215    0.021909793950950787
29  0.04116414699054414     0.023835853009455864
30  0.0699264124551481      0.004926412455148091
Yes
31  0.03955781375136214     0.02544218624863786
32  0.04484686609686611     0.02015313390313389
33  0.040189933523266844    0.02481006647673316
34  0.0423529411764706      0.022647058823529402
35  0.057080745341614905    0.007919254658385097
36  0.04659822866344606     0.018401771336553942
37  0.04289961898657551     0.022100381013424493
38  0.04259661010233092     0.02240338989766908
39  0.04131617175095436     0.023683828249045642
40  0.06471861471861473     2.813852813852713E-4
Yes
41  0.03911608289015221     0.02588391710984779
42  0.042827306361140946    0.022172693638859056
43  0.04171086631306949     0.02328913368693051
44  0.04420520999468369     0.02079479000531631
```

```
45  0.05933723196881091     0.0056627680311890916
46  0.043216523833026745    0.021783476166973258
47  0.04443419771790762     0.020565802282092385
48  0.04750476579520697     0.017495234204793035
49  0.04348406029078297     0.02151593970921703
50  0.05988562091503265     0.005114379084967355
51  0.03838908112264514     0.02661091887735486
52  0.04560708898944191     0.019392911010558095
53  0.04540325564187938     0.019596744358120623
54  0.04484126984126985     0.020158730158730154
55  0.0474025974025974      0.017597402597402605
56  0.04287840136054421     0.022121598639455793
57  0.04095655806182121     0.02404344193817879
58  0.04408235442718201     0.020917645572817993
59  0.037722729248152974    0.02727727075184703
60  0.06880341880341878     0.003803418803418779
Yes
61  0.040749414519906324    0.024250585480093678
62  0.040027177123951305    0.024972822876048698
63  0.04212454212454211     0.022875457875457895
64  0.04407051282051279     0.020929487179487213
65  0.0595660749506903      0.005433925049309701
66  0.042818292818292775    0.022181707181707228
67  0.039331315450718426    0.025668684549281576
68  0.04180378445084326     0.02319621554915674
69  0.041518867605824114    0.023481132394175888
70  0.06635031635031634     0.0013503163503163357
Yes
71  0.04187596441117566     0.023124035588824345
72  0.04680134680134677     0.018198653198653235
73  0.03897882938978825     0.026021170610211752
74  0.03963963963963961     0.025360360360360393
75  0.0499393939393939      0.0150606060606061
76  0.040749601275917015    0.024250398724082987
77  0.04474616292798109     0.020253837072018915
78  0.04417249417249414     0.020827505827505866
79  0.040378468226569465    0.024621531773430537
80  0.06429292929292925     7.070707070707533E-4
Yes
81  0.04372116223968072     0.02127883776031928
82  0.03818674550381865     0.026813254496181355
83  0.04016064257028112     0.024839357429718885
84  0.039153439153439135    0.025846560846560868
....
```

P.S. - “Yes” is being printed after every 10 lines which signifies that the length of the key is 10. Also in the beginning, 1 has not been considered as the length of the key, since it is quite improbable and intuitive. Here I have selected the *diff*(Refer code) value as 0.004

just to signify the fact that at the key length separation indices, the IOC values change abruptly and hence their difference from the standard 0.065 is minimum(As is mentioned in the slides). Hence, the length of the key is 10.

Second, we need to find the key value which is done using Kasiski's Method, the C++ code of which is as follows:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string
str1="czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrhzyczgn
waazororioflqtfrvtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqioxiazoge
vzhtdrtmnpvretngkokpokiygnlxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiqjiygzt
wyqakocagpyebvktscrgnlzlcocdckitmfyochbpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvcfcfgno
iamyctvmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnywyrohlngalitkudiazzrknjelrrnhumeytlnqbhxia
jrpafhhzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfzdvtgmevgajkbaloaztwyqakegeeuy
jivjtzhnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpmndydqfzavefviltqgmlc
ubhvjrripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjaxapacempu
mpucpckpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgsptbyzz
frjrflrluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfvf";
    int len = str1.size(),j=0;
    string str[len/10+1];
    for(int i=0;i<len;i+= 10)
    {
        if(len - i > 10)
            str[j] = str1.substr(i,10);
        else if(len - i < 10)
            str[j] = str1.substr(i,len - i);
        j++;
    }
    string str2[10];
    for(int i=0;i<10;i++)
    {
        for(int j=0;j<len/10+1;j++)
        {
            str2[i]+= str[j][i];
        }
    }
    for(int i=0;i<10;i++)
    {
        vector<pair<int,char> > vct;
        for(int j=0;j<26;j++)
        {
            vct.push_back(make_pair(0,(char)(j+'a')));
        }
        for(int k=0;k<str2[i].size();k++)
        {
```

```cpp
            int meow=(int)(str2[i][k]-'a');
            vct[meow].first++;
        }

        sort(vct.rbegin(),vct.rend());  //sorting the elements of the pair w.r.t. The first element of the pair to print the topmost recurring elements.

        cout<<"The topmost recurring elements of the "<< i+1<< " column are:"<<endl;
        for(int j=0;j<6;j++)
        {
            cout<<vct[j].second<< "\t"<<vct[j].first<<endl;
        }
    }
    cout<<str2;
    return 0;
}
```

Output: Run on IDE

```
The topmost recurring elements of the 1 column are:
t   9
i   9
e   9
o   8
a   8
r   7
The topmost recurring elements of the 2 column are:
p   16
l   10
z   8
t   7
y   6
f   5
The topmost recurring elements of the 3 column are:
n   13
a   10
e   9
t   6
o   6
s   5
The topmost recurring elements of the 4 column are:
r   13
a   10
g   8
b   7
v   6
q   5
The topmost recurring elements of the 5 column are:
b   9
t   8
```

```
n   7
l   7
a   7
p   6
The topmost recurring elements of the 6 column are:
c   11
y   10
h   10
n   7
m   7
z   5
The topmost recurring elements of the 7 column are:
z   10
v   10
r   8
k   7
e   7
u   6
The topmost recurring elements of the 8 column are:
m   12
i   10
v   8
c   8
t   6
q   5
The topmost recurring elements of the 9 column are:
r   12
q   8
v   7
f   7
a   6
y   5
The topmost recurring elements of the 10 column are:
k   10
j   10
z   8
o   8
g   8
t   6
0x7fff0c263b00
```

So, what the code does is, it basically breaks down the cipher text to an array of strings each of which is of length of that of the key(except the last few characters), i.e. 10, and then we are left with 86 strings(here, since the length of the cipher text is 856, so 86 lines of length 10), as shown:

1	2	3	4	5	6	7	8	9	10

c	z	u	y	w	u	d	I	p	h
g	d	c	a	o	c	l	t	r	p
c	k	p	u	a	m	l	n	f	h
h	r	v	h	t	l	t	m	v	m

. . . .

. . . . 86 such rows.

Table 1

So, according to Kasiski’s method and also through mere observation, we can say that the letters in every column go through the same shift, of which we take advantage. We can apply Caesar Cipher decryption to each column to figure out the corresponding letter of the key.

Presently the key looks something like:

L1	L2	L3	L4	L5	L6	L7	L8	L9	L10

Let’s make some important observations after which we can arrive at some conclusion, which would help us to identify the letters of the Key.

1. As the output of the above mentioned code suggests, p is the most recurring element(of the second column and also of all the letters present in the Cipher text. Hence an intelligent guess would be that the letter “p” of Cipher Text maps with “e” of English language, since is statistically the most recurring letter. Hence, we can say that the key’s second character, i.e., L2 would be “p-e” = “l” Similarly, if we map “e” with the most recurring elements of all the columns we would get:

p	l	j	n	x	y	r	i	n	g

2. The next observation would be to find out all the one letter words’ index, since they can be mapped to either “a” or “i”. But first we need to find out the index and the letters. The following C++ code finds out the all the single letter words and their indices and then calculates the column to which they belong to, in Table 1.

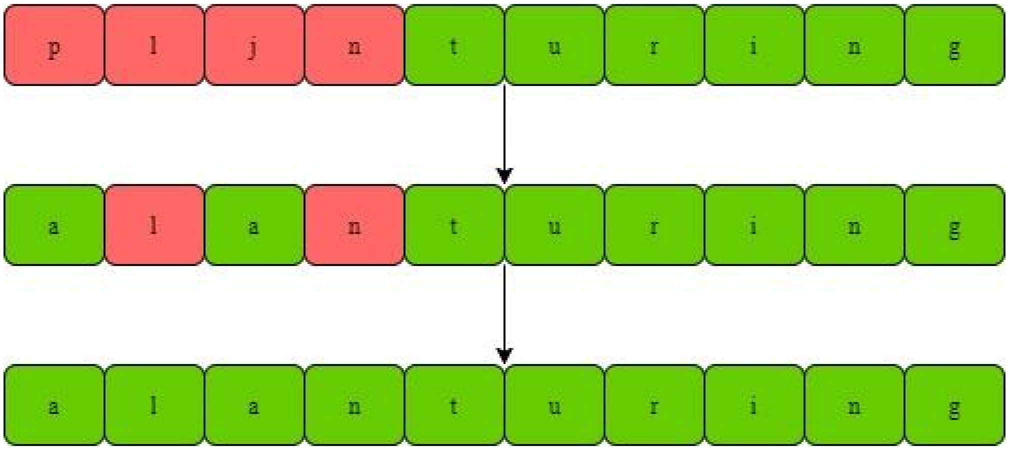
```
#include<bits/stdc++.h>
using namespace std;
int main()
```

```cpp
{
    string str1="czuyw u dipniye phgdcaocltr pckp uamlnf hh rv htltmvmyu arz oq tbicta gnrzuta zccrtt fsr hz yczgn waazoror? ioflq t frvtaare hlceo zgcsiax azdr zhp aciyrzntcp os nhumeytlnqbhx arttpnpxm rvq ioxiaz og evzh tdrtm npvre tn gkokp okiyg nl xvdbod zf gailouzs lnq tm vuczy tnfbxv if g ntnrmyvvgn cpngnlp iqjiyg ztwyqak oc a gpyebvkts crgnlzl cocd ckitmfyoc? hbp gzouz wp dvlnzvtaidh oxnnmrt a reancemye cznfvcfcf gno iamyctvmeyt zbhu iaj bft n vfvdrxlj cbgmkzhitpd onn ywyroh lngalitk udiaz zrknje? lrr nhumeytlnqbhx iaj rpafhhzvt onnoziukqore higa grbrxillvlnzk, zkcsaabmkqp bipw by fzdvtg mevgaj? kbalo a ztwyqak egee uy jivj tz hnoy diqk ies bph umpostoal? wfcyj a xapacem ugvp brecvnf. ioflq t grkuonp mndy dqfzavef? viltq g mlcubhv jrripvr bn diqk ies bph umpostoal? wfcyj a xapacem rxrznrhojtl lrpe jbfc bb otdeyy? wfcyj a xapacem pump uc pckp vjels gauk pnbe yog uyzvt vrzgetgdmq oneo vm ce iqbaycr. viltq irpagbpvtl kmprtx ziwz g spt by zzfrj rflrl? uim jk egea mbv ubyt nrrtnzdr gmznt nm scg, vadsvoy jtnbed purmzkf zhlt thpvza uuc nrnlfvf?";
    int len = str1.size(),j=0;
    int kount=0;
    for(int i = 0; i < len; i++)
    {
        if(!((str1[i]) >= 'a' && (str1[i]) <='z'))
        {
            kount++;
            if(str1[i] == ' ')
            {
                if((str1[i+1] != ' ') && (str1[i+1]) >= 'a' && (str1[i+1]) <='z' && (str1[i+2]) == ' ')
                {
                   cout<< str1[i+1] <<"\t"<<(i+2-kount)%10<< endl;
                   kount++;
                   i+=3;
                }
            }
        }
    }
    return 0;
}
```

Output: Run on IDE

```
• u   6
• t   5
• g   0
• a   3
• a   1
• n   4
• a   3
• a   1
• t   5
• g   0
• a   1
• a   1
• g   0
```

3. After intuitive Monographic mapping of each of these letters to “a”, we get a more meaningful key:



p l j n t u r i n g

a l a n t u r i n g

a l a n t u r i n g

Transition Diagram

Hence the final key, most probably, would be “alanturing”.

STEP 4

In this step we need to do two things:

1. Decrypt the cipher text using the key value(“alanturing”), which we just found in the previous step. The following is the JAVA code which does the same.

```java
public class VigenereCipher
{

    public static String decrypt(String text, final String key)
    {
        String res = "";
        text = text.toLowerCase();
        for (int i = 0, j = 0; i < text.length(); i++)
        {
            char c = text.charAt(i);
            if (c < 'a' || c > 'z')
                continue;
            res += (char) ((c - key.charAt(j) + 26) % 26 + 'a');
            j = ++j % key.length();
        }
        return res;
    }

    public static void main(String[] args)
    {
        String key = "alanturing";
        String message = 
"czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrhzyczgnwaazo
rorioflqtfrvtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqioxiazogevzhtd
rtmnpvretngkokpokiygnlxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiqjiygztwyqak
ocagpyebvktscrgnlzlcocdckitmfyochbpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvcfcfgnoiamyc
tvmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnywyrohlngalitkudiazzrknjelrrnhumeytlnqbhxiajrpaf
hhzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfzdvtgmevgajkbaloaztwyqakegeeuyjivjt
zhnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpmndydqfzavefviltqgmlcubhvj
rripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjaxapacempumpucp
ckpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgsptbyzzfrjrf
lrluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfvf";
        System.out.println("String: " + message);
        System.out.println("Encrypted message input: " + message);
        System.out.println("Decrypted message output: " + decrypt(message, key));
    }
}
```

Output: Run on IDE

```
● couldamachinecommunicatewithhumansonanunlimitedsetoftopicsthroughfluentuseofhumanlangua
gecouldalanguageusingmachinegivetheappearanceofunderstandingsentencesandcomingupwithide
aswhileintruthbeingasdevoidofthoughtsandasemptyinsideasanineteenthcenturyaddingmachineo
ratwentiethcenturywordprocessorhowmightwedistinguishbetweenagenuinelyconsciousandintell
igentmindandbutacleverlyconstructedbuthollowlanguageusingfacadeareunderstandingandreaso
ningincompatiblewithmaterialisticmechanisticviewoflivingbeingscouldamachineeverbesaidto
havemadeitsowndecisionscouldamachinehavebeliefscouldamachinemakemistakescouldamachinebe
lieveitmadeitsowndecisionscouldamachineerroneouslyfreewilltoitselfcouldamachinecomeupwi
thideasthathavenotbeingprogrammedintoitinadvancecouldcreativelyemergefromasetoffixedrul
esareweeventhemostcreativeamongusbutpassiveslavesphysicsthatgovernourneurons
```

2. But, the decryption process is not over yet, since we need to insert the special characters right in place as they were originally in the Cipher Text. The following C++ code does the same.

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

int main()
{
    string str1="czuyw u dipniye phgdcaocltr pckp uamlnf hh rv htltmvmyu arz oq tbicta
gnrzuta zccrtt fsr hz yczgn waazoror? ioflq t frvtaare hlceo zgcsiax azdr zhp
aciyrzntcp os nhumeytlnqbhx arttpnpxm rvq ioxiaz og evzh tdrtm npvre tn gkokp okiyg nl
xvdbod zf gailouzs lnq tm vuczy tnfbxv if g ntnrmyvvgn cpngnlp iqjiyg ztwyqak oc a
gpyebvkts crgnlzl cocd ckitmfyoc? hbp gzouz wp dvlnzvtaidh oxnnmrt a reancemye
cznfvcfcf gno iamyctvmeyt zbhu iaj bft n vfvdrxlj cbgmkzhitpd onn ywyroh lngalitk udiaz zrknje? lrr nhumeytlnqbhx iaj rpafhhzvt onnoziukqore higa grbrxillvlnzk, zkcsaabmkqp bipw by fzdvtg mevgaj? kbalo a ztwyqak egee uy jivj tz hnoy diqk ies bph umpostoal? wfcyj a xapacem ugvp brecvnf. ioflq t grkuonp mndy dqfzavef? viltq g mlcubhv jrripvr bn diqk ies bph umpostoal? wfcyj a xapacem rxrznrhojtl lrpe jbfc bb otdeyy? wfcyj a xapacem pump uc pckp vjels gauk pnbe yog uyzvt vrzgetgdmq oneo vm ce iqbaycr. viltq irpagbpvtl kmprtx ziwz g spt by zzfrj rflrl? uim jk egea mbv ubyt nrrtnzdr gmznt nm scg, vadsvoy jtnbed purmzkf zhlt thpvza uuc nrnlfvf?";  //str1 represents the given cipher text with special characters included.
    string str2="couldamachinecommunicatewithhumansonanunlimitedsetoftopicsthroughfluentuseofhumanlanguagecouldalanguageusingmachinegivetheappearanceofunderstandingsentencesandcomingupwithideaswhileintruthbeingasdevoidofthoughtsandasemptyinsideasanineteenthcenturyaddingmachineoratwentiethcenturywordprocessorhowmightwedistinguishbetweenagenuinelyconsciousandintelligentmindandbutacleverlyconstructedbuthollowlanguageusingfacadeareunderstandingandreasoningincompatiblewithmaterialisticmechanisticviewoflivingbeingscouldamachineeverbesaidtohavemadeitsowndecisionscouldamachinehavebeliefscouldamachinemakemistakescouldamachinebelieveitmadeitsowndecisionscouldamachineerroneouslyfreewilltoitselfcouldamachinecomeupwithideasthathavenotbeingprogrammedintoitinadvancecouldcreativelyemergefromasetoffixedrulesareweeventhemostcreativeamongusbutpassiveslavesphysicsthatgovernourneurons";
//str2 represents the decrypted text without special characters.

    string str3;
    for(int i=0;i<str2.size()+1;i++)
    {
        if(str1[i]<'a' or str1[i]>'z')
        {
            str3 = str1[i];
            str2.insert(i,str3);
        }
    }
    cout<<str2;
    return 0;
}
```

Final Output:

Run on IDE

```
• could a machine communicate with humans on an unlimited set of topics through fluent
  use of human language? could a language using machine give the appearance of
  understanding sentences and coming up with ideas while in truth being as devoid of
  thoughts and as empty inside as a nineteenth century adding machine or a twentieth
  century word processor? how might we distinguish between a genuinely conscious and
  intelligent mind and but a cleverly constructed but hollow language using facade? are
  understanding and reasoning incompatible with materialistic, mechanistic view of living
  beings? could a machine ever be said to have made its own decisions? could a machine
  have beliefs. could a machine make mistakes? could a machine believe it made its own
  decisions? could a machine erroneously free will to itself? could a machine come up
  with ideas that have not being programmed into it in advance. could creatively emerge
  from a set of fixed rules? are we even the most creative among us but, passive slaves
  physics that govern our neurons?
```

________END______