Assignment Report

Name: Aditya Sundarlal Darak

ID: 2014B4A30807P

Given encrypted text:

czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrhzyczgnwaazororioflqtfrvtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqioxiazogevzhtdrtmnpvretngkokpokiygnlxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiqjiygztwyqakocagpyebvktscrgnlzlcocdckitmfyochbpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvcfcfgnoiamyctvmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnywyrohlngalitkudiazzrknjelrrnhumeytlnqbhxiajrpafhhzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfzdvtgmevgajkbaloaztwyqakegeeuyjivjtzhnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpmndydqfzavefviltqgmlcubhvjrripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjaxapacempumpucpckpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgsptbyzzfrjrflrluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfvf

Decrypted Text:

Couldamachinecommunicatewithhumansonanunlimitedsetoftopicsthroughfluentuseofhumanlanguagecouldalanguageusingmachinegivetheappearanceofunderstandingsentencesandcomingupwithideaswhileintruthbeingasdevoidofthoughtsandasemptyinsideasanineteenthcenturyaddingmachineoratwentiethcenturywordprocessorhowmightwedistinguishbetweenagenuinelyconsciousandintelligentmindandbutacleverlyconstructedbuthollowlanguageusingfacadeareunderstandingandreasoningincompatiblewithmaterialisticmechanisticviewoflivingbeingscouldamachineeverbesaidtohavemadeitsowndecisionscouldamachinehavebeliefscouldamachinemakemistakescouldamachinebelieveitmadeitsowndecisionscouldamachineerroneouslyfreewilltoitselfcouldamachinecomeupwithideasthathavenotbeingprogrammedintoitinadvancecouldcreativelyemergefromasetoffixedrulesareweeventhemostcreativeamongusbutpassiveslavesphysicsthatgovernourneurons

Procedure adopted to find the decrypted text:

1) Finding type of cipher:

English language has index of coincidence value of 0.065. I used the following code to find the Ic of given cipher text.

```
#include<iostream>

#include<string>

#include<vector>

#include<map>

#include<iterator>

#include<cmath>

using namespace std;

int main(){

            string s
("czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrhzyczgnwaazororioflqtfr
vtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqioxiazogevzhtdrtmnpvretngkokpokiygn
lxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiqjiygztwyqakocagpyebvktscrgnlzlcocdckitmfyoch
bpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvcfcfgnoiamyctvmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnyw
yrohlngalitkudiazzrknjelrrnhumeytlnqbhxiajrpafhhzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfz
dvtgmevgajkbaloaztwyqakegeeuyjivjtzhnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpm
ndydqfzavefviltqgmlcubhvjrripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjax
apacempumpucpckpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgsptbyz
zfrjrflrluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfvf");

  map<char, float> map1;

  double num = 0;

  int i=0;

  while(i<s0.length()){

                        map1[s[i]]++;

                        i++;

                }


  for(map<char, float>::iterator t = map1.begin();t!=map1.end();++t){

                if(t->second!=0 || t->second!=1)
```

```
                            num = num + t->second*(t->second-1)/2;
  }
  int n=s0.length();
  double den=n*(n-1)/2;
  cout<<num/den;
  return 0;
}
```

The index of coincidence value returned by the above code is 0.04225565 which is much lesser than 0.065.

The above result imply that given encryption is not done using substitution cipher or shift cipher because even if the message is substituted or shifted, it's property remains the same and hence the index of coincidence value.

Hence, the encryption type used to convert plain text into cipher text is vigenere cipher.

Finding the key length

To find the key length of vigenere cipher, we again use the standard Index of Coincidence 0.065. Here we find the value of m such that if we consider sequences $\{y_1,y_{m+1,........}\}$, $\{y_2,y_{m+2,........}\}$, … , $\{y_{m-1},y_{2m-1,........}\}$, we find the Index of coincidence value for all such sequences. If for any given m, all these sequence yields index of coincidence of value near to 0.065, then the given m is our key length.

I used following code to find the key length of vigenere cipher:

```
#include<iostream>
#include<string>
#include<vector>
#include<map>
#include<iterator>
#include<cmath>
using namespace std;
vector<float> ioc(string s, int m){
        vector<float> ic;
        double n = s.length();
        int i=0,j;
```

```
    while(i<m){

        j = i;

        int p=0;

        map<char, float> map1;

        while(j<s.length())

        {

            map1[s[j]]++;

            j+=m;

            p++;

        }

        double value = 0;

        for(map<char, float>::iterator t = map1.begin();t!=map1.end();++t){

            if(t->second!=0 || t->second!=1)

                value = value + t->second*(t->second-1)/(p*(p-1));

        }

        ic.push_back(value);

        i++;

    }

    return ic;

}

int main()

{

    string s
("czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrhzyczgnwaazororioflqtfr
vtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqioxiazogevzhtdrtmnpvretngkokpokiygn
lxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiqjiygztwyqakocagpyebvktscrgnlzlcocdckitmfyoch
bpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvcfcfgnoiamyctvmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnyw
yrohlngalitkudiazzrknjelrrnhumeytlnqbhxiajrpafhhzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfz
dvtgmevgajkbaloaztwyqakegeeuyjivjtzhnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpm
ndydqfzavefviltqgmlcubhvjrripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjax
apacempumpucpckpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgsptbyz
zfrjrflrluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfvf");

    vector<float> ic;

    vector<float> sqrtmean;
    for(int m=1;m<s.length()/2;m++){
        ic= ioc(s, m);
        float value=0;
        for(int i=0;i<ic.size();i++){
            value=value+abs(ic[i]-0.065)*abs(ic[i]-0.065);
        }
        value= sqrt(value)/ic.size();
        sqrtmean.push_back(value);
    }
    int minval=0;
    for(int i=0;i<sqrtmean.size();i++){
        if(sqrtmean[minval]>sqrtmean[i]){
            minval=i;
        }
    }
    cout<<minval+1;
    return 0;
}
```

I have used square root mean for finding the value of m so as to find the sequence with minimum deviation from 0.065.

The result of this code is 10 which imply that key length is 10.

Finding the key:

In this part, we will use the probabilities of characters in English language to calculate chi-squared statistics for every sequence $\{y_1,y_{m+1,.......}\}$, $\{y_2,y_{m+2,........}\}$, … , $\{y_{m-1},y_{2m-1,.......}\}$ and for every sequence, we will find a character in English language which gives minimum chi-square value which will ultimately give us our key.

Following code gives the exact key corresponding to key length found in previous section:

```
#include<iostream>
#include<string>
#include<vector>
```

```
#include<map>

#include<iterator>

#include<cmath>

using namespace std;

char kap(char a,char b){

    if(a>b){

        return 26+b-a;

    }

    return b-a;

}

char findchar(string s1,int start, int m){

    int n=s1.length();

    vector <int> psivalue;

    int k=start;

    vector<char> s2;

    while(k<n){

        s2.push_back(s1[k]);

        k=k+m;

    }

    for(char i='a';i<'z'+1;i++){

        vector<char> str;

        int r=s2.size();

        k=0;

        while(k<r){

            char b=97+ kap(i,s2[k]);

            str.push_back(b);

            k++;

        }

        float d[26]={0.08167,0.01492,0.02782, 0.04253, 0.12702, 0.0228,0.02015,0.06094,0.06996, 0.00153, 0.00772,0.04025,0.02406,0.06749,0.07507,0.01929,0.00095,0.05987,0.06327,0.09056,0.02758,0.00978,0.02362,0.00150,0.01974,0.00074};
        float psi=0;

        map<char, float> map1;

        k=0;

        while(k<r){

            map1[str[k]]++;

            k++;

        }

        for(map<char, float>::iterator it = map1.begin();it!=map1.end();++it)

                                {

                                                char b=it->first;

                                                //a=a-'a';

                                                int p=int(b)-97;

                                                float prob=r*d[p];

                                                psi=psi+(it->second-prob)*(it->second-prob)/prob;

        }

        psivalue.push_back(psi);

    }

    int minval=0;

    for(int i=1;i<26;i++){

        if(psivalue[i]<psivalue[minval]){

            minval=i;

        }

    }

    char u=char(97+minval);

    return u;

}

int main(){

                string s
("czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrhzyczgnwaazororioflqtfr
vtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqioxiazogevzhtdrtmnpvretngkokpokiygn
lxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiqjiygztwyqakocagpyebvktscrgnlzlcocdckitmfyoch
bpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvcfcfgnoiamyctvmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnyw
```

```
yrohlngalitkudiazzrknjelrrnhumeytlnqbhxiajrpafhhzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfz
dvtgmevgajkbaloaztwyqakegeeuyjivjtzhnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpm
ndydqfzavefviltqgmlcubhvjrripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjax
apacempumpucpckpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgsptbyz
zfrjrflrluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfvf");

        int keylength =10;

        vector<char> key;

        for(int i=0;i<keylength;i++){

                key.push_back(findchar(s,i,keylength));

                cout<<key[i];

        }

        cout<<endl;

        return 0;

}
```

The given code outputs the key as “alanturing”.

Appendix:

Complete code for decrypting any message encrypted using vigenere cipher is as follows:

```
#include<iostream>
#include<string>
#include<vector>
#include<map>
#include<iterator>
#include<cmath>
using namespace std;

char kap(char a,char b){
    if(a>b){
        return 26+b-a;
    }
    return b-a;
}

vector<float> ioc(string s, int m){
        vector<float> ic;
        double n = s.length();
        int i=0,j;
        while(i<m){
                j = i;
                int p=0;
                map<char, float> map1;
                while(j<s.length())
                {
                        map1[s[j]]++;
                        j+=m;
```

```
            p++;
        }
        double value = 0;
        for(map<char, float>::iterator t = map1.begin();t!=map1.end();++t)
        {
            if(t->second!=0 || t->second!=1)
                value = value + t->second*(t->second-1)/(p*(p-1));
        }
        ic.push_back(value);
        i++;
    }
    return ic;
}

char findchar(string s1,int start, int m){
    int n=s1.length();
    vector <int> psivalue;
    int k=start;
    vector<char> s2;
    while(k<n){
        s2.push_back(s1[k]);
        k=k+m;
    }
    for(char i='a';i<'z'+1;i++){
        vector<char> str;
        int r=s2.size();
        k=0;
        while(k<r){
            char b=97+ kap(i,s2[k]);
            str.push_back(b);
            k++;
        }
        float d[26]={0.08167,0.01492,0.02782, 0.04253, 0.12702, 0.0228,0.02015,0.06094,0.06996, 0.00153,
0.00772,0.04025,0.02406,0.06749,0.07507,0.01929,0.00095,0.05987,0.06327,0.09056,0.02758,0.00978,0.02
362,0.00150,0.01974,0.00074};
        float psi=0;
        map<char, float> map1;
        k=0;
        while(k<r){
            map1[str[k]]++;
            k++;
        }
        for(map<char, float>::iterator it = map1.begin();it!=map1.end();++it)
                                {
                                                char b=it->first;
                                                //a=a-'a';
                                                int p=int(b)-97;
                                                float prob=r*d[p];
                                                psi=psi+(it->second-prob)*(it->second-prob)/prob;
        }
        psivalue.push_back(psi);
    }
    int minval=0;
    for(int i=1;i<26;i++){
        if(psivalue[i]<psivalue[minval]){
            minval=i;
        }
    }
    char u=char(97+minval);
    return u;
}
```

```
int main(){

        string s
("czuywudipniyephgdcaocltrpckpuamlnfhhrvhtltmvmyuarzoqtbictagnrzutazccrttfsrhzyczgnwaazororioflqtfr
vtaarehlceozgcsiaxazdrzhpaciyrzntcposnhumeytlnqbhxarttpnpxmrvqioxiazogevzhtdrtmnpvretngkokpokiygn
lxvdbodzfgailouzslnqtmvuczytnfbxvifgntnrmyvvgncpngnlpiqjiygztwyqakocagpyebvktscrgnlzlcocdckitmfyoch
bpgzouzwpdvlnzvtaidhoxnnmrtareancemyecznfvcfcfgnoiamyctvmeytzbhuiajbftnvfvdrxljcbgmkzhitpdonnyw
yrohlngalitkudiazzrknjelrrnhumeytlnqbhxiajrpafhhzvtonnoziukqorehigagrbrxillvlnzkzkcsaabmkqpbipwbyfz
dvtgmevgajkbaloaztwyqakegeeuyjivjtzhnoydiqkiesbphumpostoalwfcyjaxapacemugvpbrecvnfioflqtgrkuonpm
ndydqfzavefviltqgmlcubhvjrripvrbndiqkiesbphumpostoalwfcyjaxapacemrxrznrhojtllrpejbfcbbotdeyywfcyjax
apacempumpucpckpvjelsgaukpnbeyoguyzvtvrzgetgdmqoneovmceiqbaycrviltqirpagbpvtlkmprtxziwzgsptbyz
zfrjrflrluimjkegeambvubytnrrtnzdrgmzntnmscgvadsvoyjtnbedpurmzkfzhltthpvzauucnrnlfvf");

        vector<float> ic;

        vector<float> sqrtmean;

        for(int m=1;m<s.length()/2;m++){

                ic= ioc(s, m);

                float value=0;

                for(int i=0;i<ic.size();i++){

      value=value+abs(ic[i]-0.065)*abs(ic[i]-0.065);

                }

                value= sqrt(value)/ic.size();

                sqrtmean.push_back(value);

        }

        int minval=0;

        for(int i=0;i<sqrtmean.size();i++){

    if(sqrtmean[minval]>sqrtmean[i]){

      minval=i;

    }

        }

        cout<<minval+1<<endl;

        vector<char> key;

        for(int i=0;i<minval+1;i++){

      key.push_back(findchar(s,i,minval+1));

      cout<<key[i];
```

```
        }
        cout<<endl;
        for(int i=0;i<s.length();){
    for(int j=0;j<10;j++){
      char p=97+kap(key[j],s[i]);
      cout<<p;
      i++;
    }
        }
        cout<<endl;
        return 0;
}
```