INTRODUCTION TO CRYPTOGRAPHY – LAB 3

B.Tech. Computer Science and Engineering (Cybersecurity)

Name: Anish Sudhan Nair	Roll No.: K041
Batch: K2/A2	Date of performance: 12/12/2021

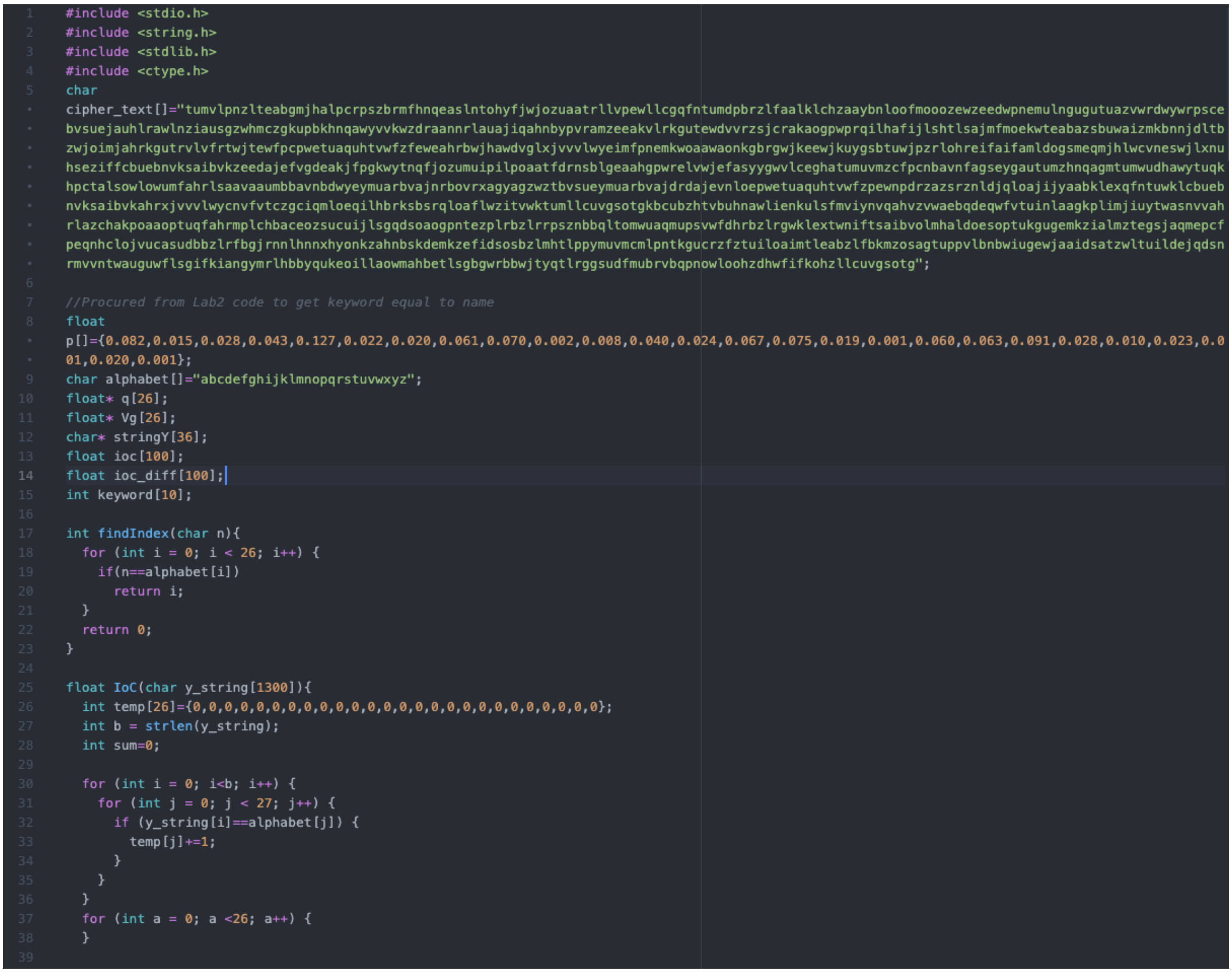
Aim: To perform cryptanalysis of the Vigenere cipher using Friedman test

Code:

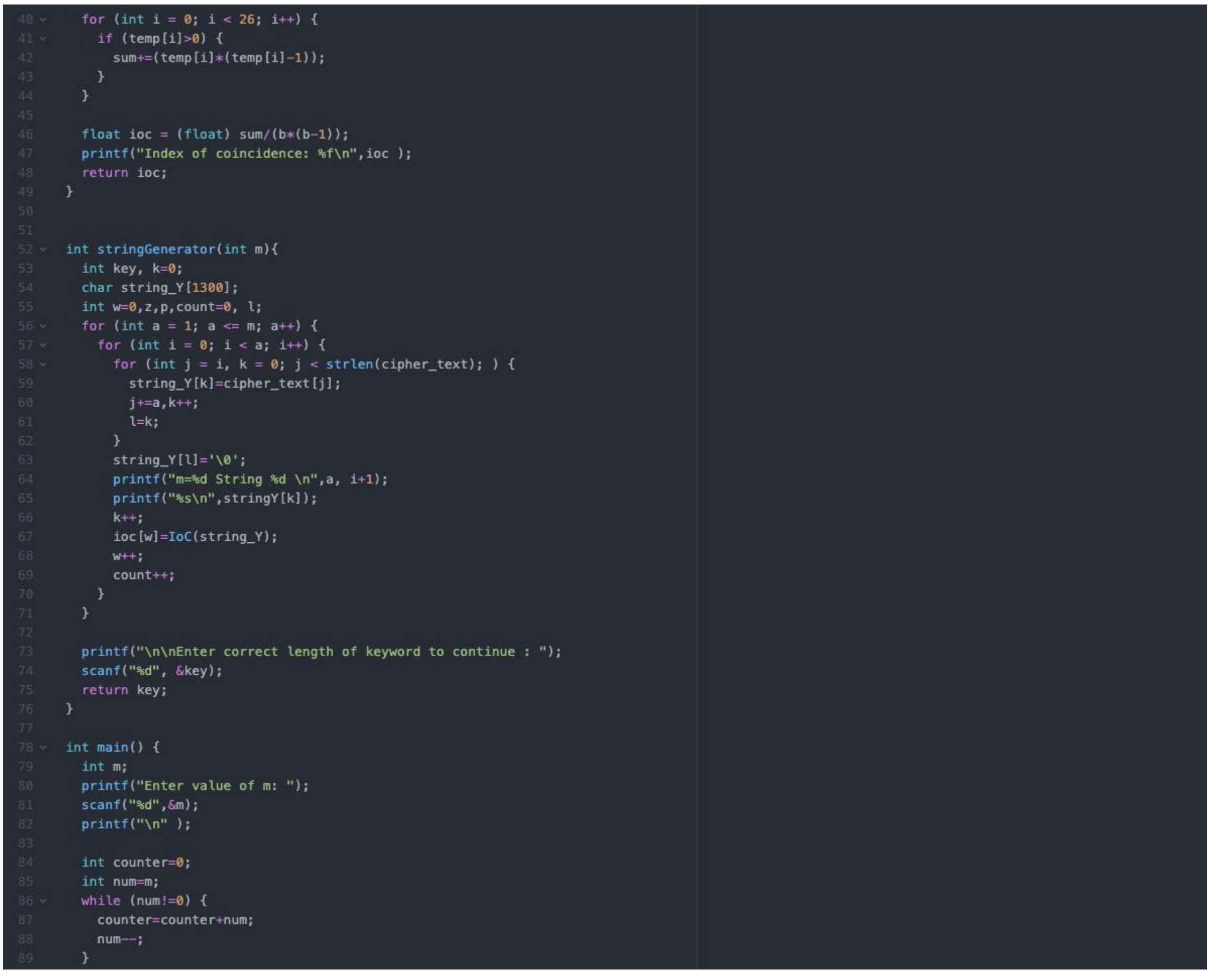
Language: C

Editor: Atom

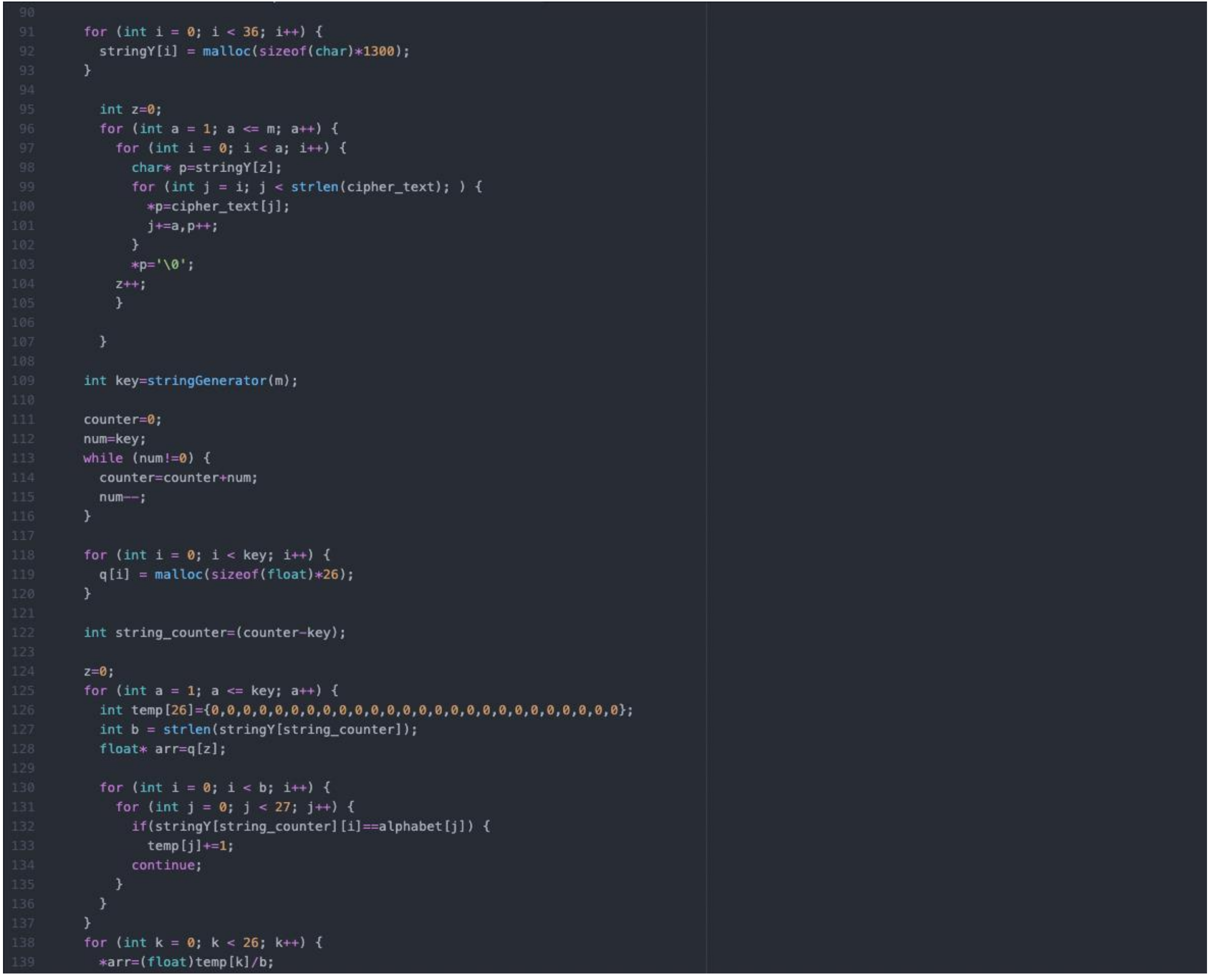
Compiler: clang/ZSH



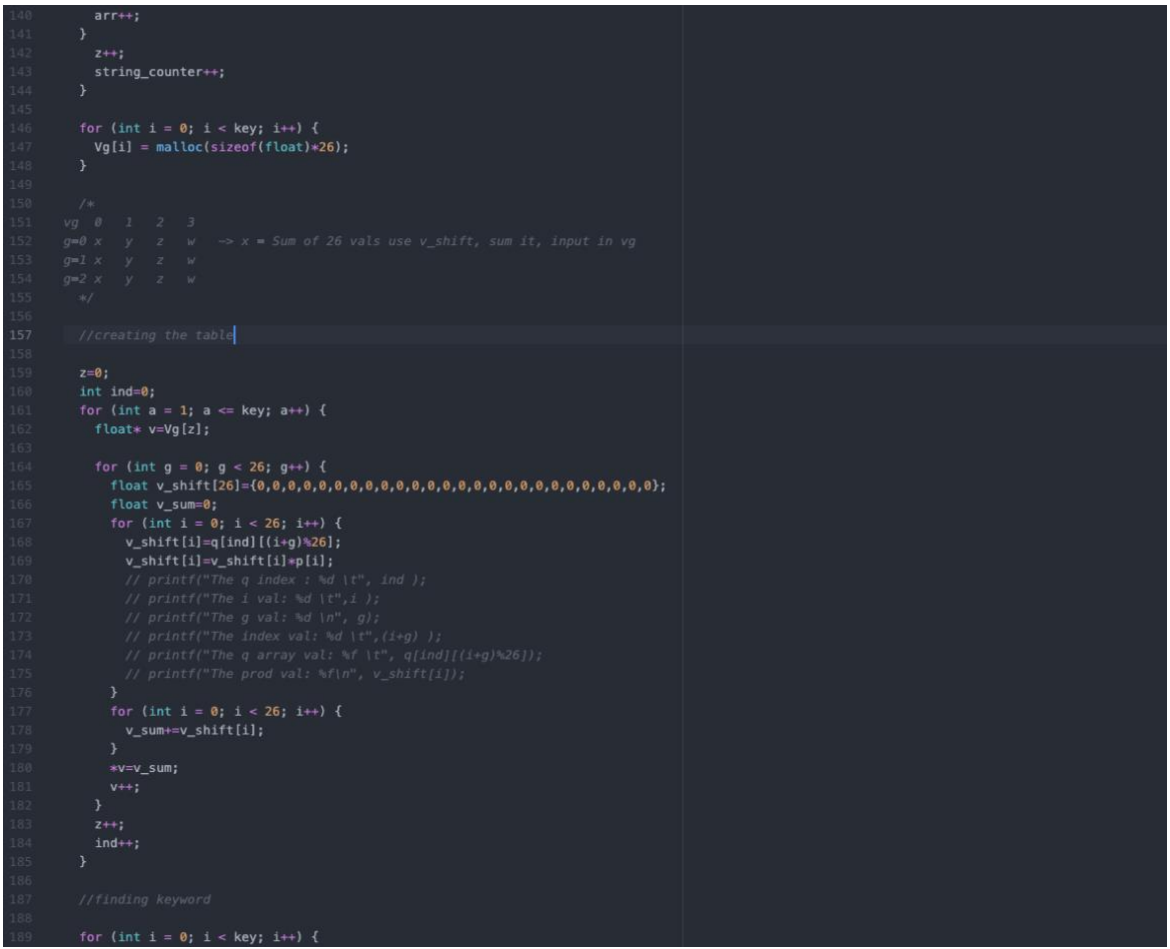
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
char
cipher_text[]="tumvlpnzlteabgmjhalpcrpszbrmfhnqeaslntohyfjwjozuaatrllvpewllcgqfntumdpbrzlfaalklchzaaybnloofmooozewzeedwpnemulngugutuazvwrdwywrpsce
bvsuejauhlrawlnziausgzwhmczgkupbkhnqawyvvkwzdraannrlauajiqahnbypvramzeeakvlrkgutewdvvrzsjcrakaogpwprqilhafijlshtlsajmfmoekwteabazsbuwaizmkbnnjdltb
zwjoimjahrkgutrvlvfrtwjtewfpcpwetuaquhtvwfzfeweahrbwjhawdvglxjvvvlwyeimfpnemkwoaawaonkgbrgwjkeewjkuygsbtuwjpzrlohreifaifamldogsmeqmjhlwcvneswjlxnu
hseziffcbuebnvksaibvkzeedajefvgdeakjfpgkwytnqfjozumuipilpoaatfdrnsblgeaahgpwrelvwjefasyygwvlceghatumuvmzcfpcnbavnfagseygautumzhnqagmtumwudhawytuqk
hpctalsowlowumfahrlsaavaaumbbavnbdwyeymuarbvajnrbovrxagyagzwztbvsueymuarbvajdrdajevnloepwetuaquhtvwfzpewnpdrzazsrznldjqloajijyaabklexqfntuwklcbueb
nvksaibvkahrxjvvvlwycnvfvtczgciqmloeqilhbrksbsrqloaflwzitvwktumllcuvgsotgkbcubzhtvbuhnawlienkulsfmviynvqahvzvwaebqdeqwfvtuinlaagkplimjiuytwasnvvah
rlazchakpoaaoptuqfahrmplchbaceozsucuijlsgqdsoaogpntezplrbzlrrpsznbbqltomwuaqmupsvwfdhrbzlrgwklextwniftsaibvolmhaldoesoptukgugemkzialmztegsjaqmepcf
peqnhclojvucasudbbzlrfbgjrnnlhnnxhyonkzahnbskdemkzefidsosbzlmhtlppymuvmcmlpntkgucrzfztuiloaimtleabzlfbkmzosagtuppvlbnbwiugewjaaidsatzwltuildejqdsn
rmvvntwauguwflsgifkiangymrlhbbyqukeoillaowmahbetlsgbgwrbbwjtyqtlrggsudfmubrvbqpnowloohzdhwfifkohzllcuvgsotg";

//Procured from Lab2 code to get keyword equal to name
float
p[]={0.082,0.015,0.028,0.043,0.127,0.022,0.020,0.061,0.070,0.002,0.008,0.040,0.024,0.067,0.075,0.019,0.001,0.060,0.063,0.091,0.028,0.010,0.023,0.0
01,0.020,0.001};
char alphabet[]="abcdefghijklmnopqrstuvwxyz";
float* q[26];
float* Vg[26];
char* stringY[36];
float ioc[100];
float ioc_diff[100];
int keyword[10];

int findIndex(char n){
  for (int i = 0; i < 26; i++) {
    if(n==alphabet[i])
      return i;
  }
  return 0;
}

float IoC(char y_string[1300]){
  int temp[26]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
  int b = strlen(y_string);
  int sum=0;

  for (int i = 0; i<b; i++) {
    for (int j = 0; j < 27; j++) {
      if (y_string[i]==alphabet[j]) {
        temp[j]+=1;
      }
    }
  }
  for (int a = 0; a <26; a++) {
  }

```



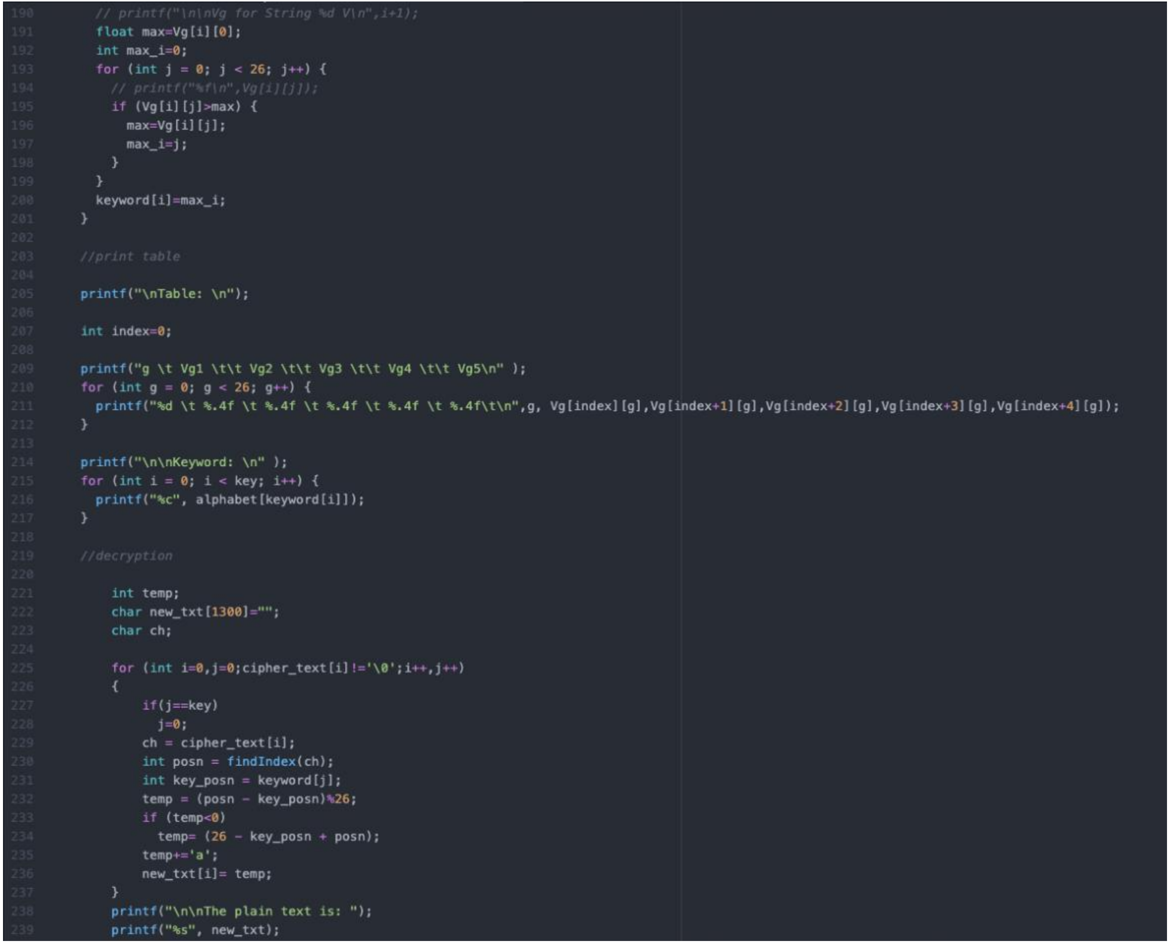
```c
  for (int i = 0; i < 26; i++) {
    if (temp[i]>0) {
      sum+=(temp[i]*(temp[i]-1));
    }
  }

  float ioc = (float) sum/(b*(b-1));
  printf("Index of coincidence: %f\n",ioc );
  return ioc;
}


int stringGenerator(int m){
  int key, k=0;
  char string_Y[1300];
  int w=0,z,p,count=0, l;
  for (int a = 1; a <= m; a++) {
    for (int i = 0; i < a; i++) {
      for (int j = i, k = 0; j < strlen(cipher_text); ) {
        string_Y[k]=cipher_text[j];
        j+=a,k++;
        l=k;
      }
      string_Y[l]='\0';
      printf("m=%d String %d \n",a, i+1);
      printf("%s\n",stringY[k]);
      k++;
      ioc[w]=IoC(string_Y);
      w++;
      count++;
    }
  }

  printf("\n\nEnter correct length of keyword to continue : ");
  scanf("%d", &key);
  return key;
}

int main() {
  int m;
  printf("Enter value of m: ");
  scanf("%d",&m);
  printf("\n" );

  int counter=0;
  int num=m;
  while (num!=0) {
    counter=counter+num;
    num--;
  }
```



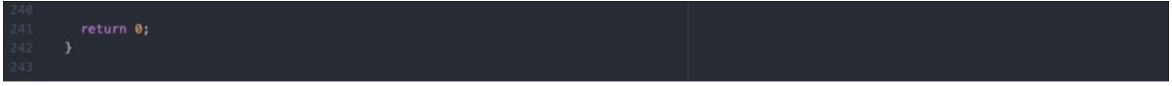
```c

  for (int i = 0; i < 36; i++) {
    stringY[i] = malloc(sizeof(char)*1300);
  }

    int z=0;
    for (int a = 1; a <= m; a++) {
      for (int i = 0; i < a; i++) {
        char* p=stringY[z];
        for (int j = i; j < strlen(cipher_text); ) {
          *p=cipher_text[j];
          j+=a,p++;
        }
        *p='\0';
      z++;
      }

    }

  int key=stringGenerator(m);

  counter=0;
  num=key;
  while (num!=0) {
    counter=counter+num;
    num--;
  }

  for (int i = 0; i < key; i++) {
    q[i] = malloc(sizeof(float)*26);
  }

  int string_counter=(counter-key);

  z=0;
  for (int a = 1; a <= key; a++) {
    int temp[26]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    int b = strlen(stringY[string_counter]);
    float* arr=q[z];

    for (int i = 0; i < b; i++) {
      for (int j = 0; j < 27; j++) {
        if(stringY[string_counter][i]==alphabet[j]) {
          temp[j]+=1;
          continue;
        }
      }
    }
  for (int k = 0; k < 26; k++) {
    *arr=(float)temp[k]/b;
```



```c
    arr++;
  }
    z++;
    string_counter++;
  }

  for (int i = 0; i < key; i++) {
    Vg[i] = malloc(sizeof(float)*26);
  }

  /*
vg  0   1   2   3
g=0 x   y   z   w   -> x = Sum of 26 vals use v_shift, sum it, input in vg
g=1 x   y   z   w
g=2 x   y   z   w
  */

  //creating the table

  z=0;
  int ind=0;
  for (int a = 1; a <= key; a++) {
    float* v=Vg[z];

    for (int g = 0; g < 26; g++) {
      float v_shift[26]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
      float v_sum=0;
      for (int i = 0; i < 26; i++) {
        v_shift[i]=q[ind][(i+g)%26];
        v_shift[i]=v_shift[i]*p[i];
        // printf("The q index : %d \t", ind );
        // printf("The i val: %d \t",i );
        // printf("The g val: %d \n", g);
        // printf("The index val: %d \t",(i+g) );
        // printf("The q array val: %f \t", q[ind][(i+g)%26]);
        // printf("The prod val: %f\n", v_shift[i]);
      }
      for (int i = 0; i < 26; i++) {
        v_sum+=v_shift[i];
      }
      *v=v_sum;
      v++;
    }
    z++;
    ind++;
  }

  //finding keyword

  for (int i = 0; i < key; i++) {
```

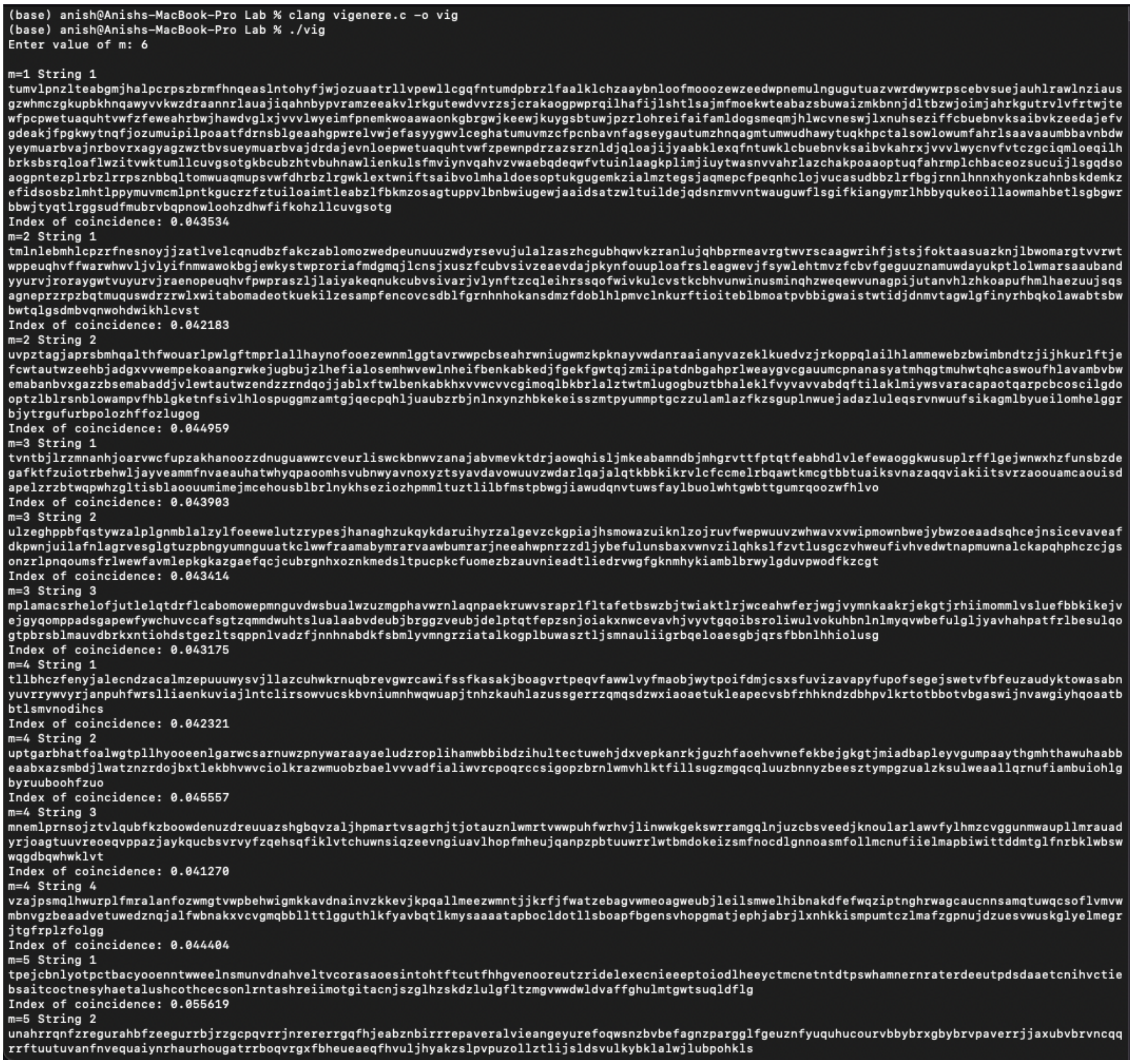


```c
    // printf("\n\nVg for String %d V\n",i+1);
    float max=Vg[i][0];
    int max_i=0;
    for (int j = 0; j < 26; j++) {
      // printf("%f\n",Vg[i][j]);
      if (Vg[i][j]>max) {
        max=Vg[i][j];
        max_i=j;
      }
    }
    keyword[i]=max_i;
  }

  //print table

  printf("\nTable: \n");

  int index=0;

  printf("g \t Vg1 \t\t Vg2 \t\t Vg3 \t\t Vg4 \t\t Vg5\n" );
  for (int g = 0; g < 26; g++) {
    printf("%d \t %.4f \t %.4f \t %.4f \t %.4f \t %.4f\t\n",g, Vg[index][g],Vg[index+1][g],Vg[index+2][g],Vg[index+3][g],Vg[index+4][g]);
  }

  printf("\n\nKeyword: \n" );
  for (int i = 0; i < key; i++) {
    printf("%c", alphabet[keyword[i]]);
  }

  //decryption

      int temp;
      char new_txt[1300]="";
      char ch;

      for (int i=0,j=0;cipher_text[i]!='\0';i++,j++)
      {
          if(j==key)
            j=0;
          ch = cipher_text[i];
          int posn = findIndex(ch);
          int key_posn = keyword[j];
          temp = (posn - key_posn)%26;
          if (temp<0)
            temp= (26 - key_posn + posn);
          temp+='a';
          new_txt[i]= temp;
      }
      printf("\n\nThe plain text is: ");
      printf("%s", new_txt);
```

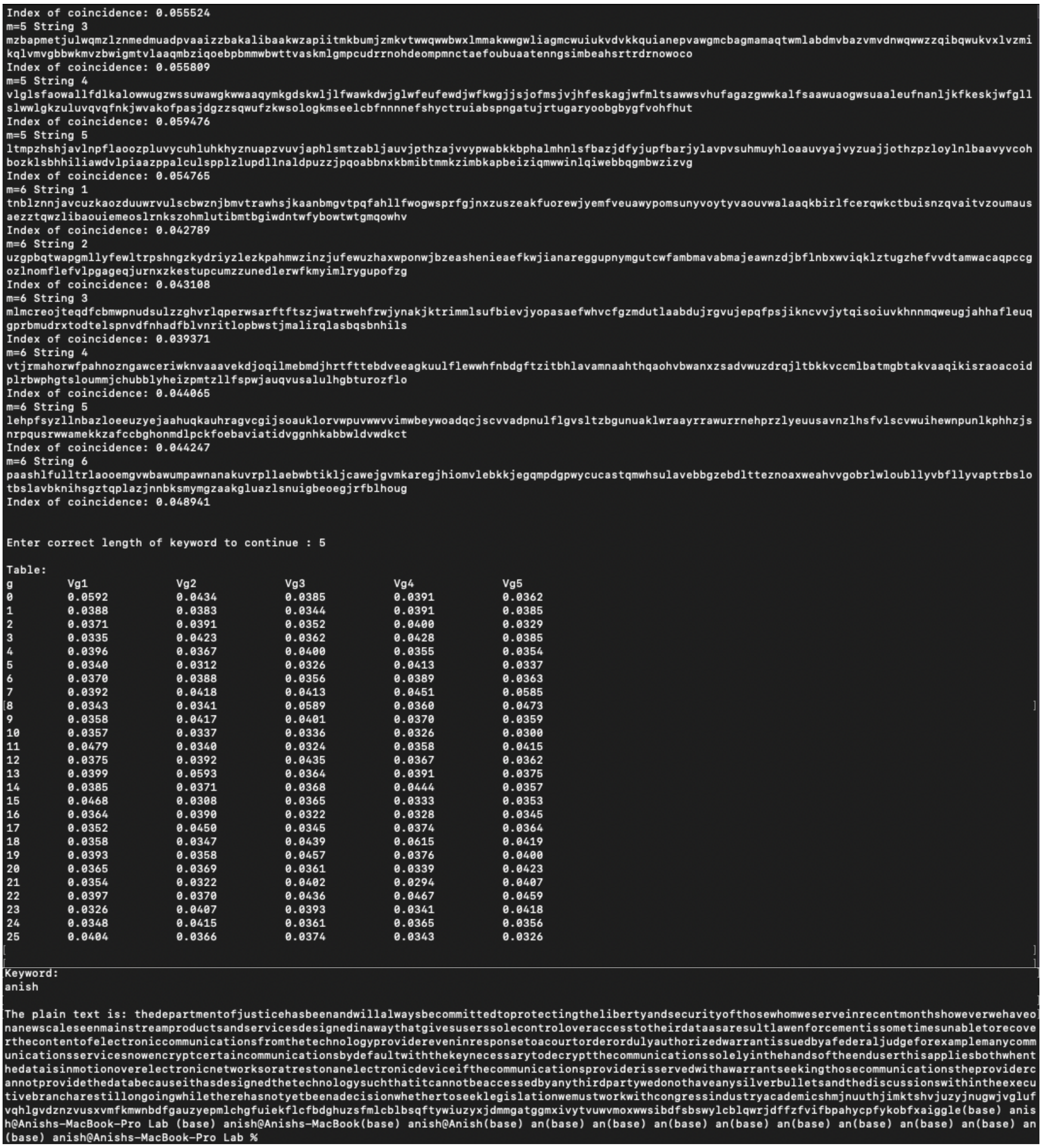


```c

  return 0;
}

```

Complete Output:



```
(base) anish@Anishs-MacBook-Pro Lab % clang vigenere.c -o vig
(base) anish@Anishs-MacBook-Pro Lab % ./vig
Enter value of m: 6

m=1 String 1
tumvlpnzlteabgmjhalpcrpszbrmfhnqeaslntohyfjwjozuaatrllvpewllcgqfntumdpbrzlfaalklchzaaybnloofmooozewzeedwpnemulngugutuazvwrdwywrpscebvsuejauhlrawlnziaus
gzwhmczgkupbkhnqawyvvkwzdraannrlauajiqahnbypvramzeeakvlrkgutewdvvrzsjcrakaogpwprqilhafijlshtlsajmfmoekwteabazsbuwaizmkbnnjdltbzwjoimjahrkgutrvlvfrtwjte
wfpcpwetuaquhtvwfzfeweahrbwjhawdvglxjvvvlwyeimfpnemkwoaawaonkgbrgwjkeewjkuygsbtuwjpzrlohreifaifamldogsmeqmjhlwcvneswjlxnuhseziffcbuebnvksaibvkzeedajefv
gdeakjfpgkwytnqfjozumuipilpoaatfdrnsblgeaahgpwrelvwjefasyygwvlceghatumuvmzcfpcnbavnfagseygautumzhnqagmtumwudhawytuqkhpctalsowlowumfahrlsaavaaumbbavnbdw
yeymuarbvajnrbovrxagyagzwztbvsueymuarbvajdrdajevnloepwetuaquhtvwfzpewnpdrzazsrznldjqloajijyaabklexqfntuwklcbuebnvksaibvkahrxjvvvlwycnvfvtczgciqmloeqilh
brksbsrqloaflwzitvwktumllcuvgsotgkbcubzhtvbuhnawlienkulsfmviynvqahvzvwaebqdeqwfvtuinlaagkplimjiuytwasnvvahrlazchakpoaaoptuqfahrmplchbaceozsucuijlsgqdso
aogpntezplrbzlrrpsznbbqltomwuaqmupsvwfdhrbzlrgwklextwniftsaibvolmhaldoesoptukgugemkzialmztegsjaqmepcfpeqnhclojvucasudbbzlrfbgjrnnlhnnxhyonkzahnbskdemkz
efidsosbzlmhtlppymuvmcmlpntkgucrzfztuiloaimtleabzlfbkmzosagtuppvlbnbwiugewjaaidsatzwltuildejqdsnrmvvntwauguwflsgifkiangymrlhbbyqukeoillaowmahbetlsgbgwr
bbwjtyqtlrggsudfmubrvbqpnowloohzdhwfifkohzllcuvgsotg
Index of coincidence: 0.043534
m=2 String 1
tmlnlebmhlcpzrfnesnoyjjzatlvelcqnudbzfakczablomozwedpeunuuuzwdyrsevujulalzaszhcgubhqwvkzranlujqhbprmeavrgtwvrscaagwrihfjstsjfoktaasuazknjlbwomargtvvrwt
wppeuqhvffwarwhwvljvlyifnmwawokbgjewkystwproriafmdgmqjlcnsjxuszfcubvsivzeaevdajpkynfouuploafrsleagwevjfsywlehtmvzfcbvfgeguuznamuwdayukptlolwmarsaauband
yyurvjroraygwtvuyurvjraenopeuqhvfpwpraszljlaiyakeqnukcubvsivarjvlynftzcqleihrssqofwivkulcvstkcbhvunwinusminqhzweqewvunagpijutanvhlzhkoapufhmlhaezuujsqs
agneprzrpzbqtmuquswdrzrwlxwitabomadeotkuekilzesampfencovcsdblfgrnhnhokansdmzfdoblhlpmvclnkurftioiteblbmoatpvbbigwaistwtidjdnmvtagwlgfinyrhbqkolawabtsbw
bwtqlgsdmbvqnwohdwikhlcvst
Index of coincidence: 0.042183
m=2 String 2
uvpztagjaprsbmhqalthfwouarlpwlgftmprlallhaynofooezewnmlggtavrwwpcbseahrwniugwmzkpknayvwdanraaianyvazeklkuedvzjrkoppqlailhlammewebzbwimbndtzjijhkurlftje
fcwtautwzeehbjadgxvvwempekoaangrwkejugbujzlhefialosemhwvewlnheifbenkabkedjfgekfgwtqjzmiipatdnbgahprlweaygvcgauumcpnanasyatmhqgtmuhwtqhcaswoufhlavambvbw
emabanbvxgazzbsemabaddjvlewtautwzendzzrndqojjablxftwlbenkabkhxvvwcvvcgimoqlbkbrlalztwtmlugogbuztbhaleklfvyvavvabdqftilaklmiywsvaracapaotqarpcbcoscilgdo
optzlblrsnblowampvfhblgketnfsivlhlospuggmzamtgjqecpqhljuaubzrbjnlnxynzhbkekeisszmtpyummptgczzulamlazfkzsguplnwuejadazluleqsrvnwuufsikagmlbyueilomhelggr
bjytrgufurbpolozhffozlugog
Index of coincidence: 0.044959
m=3 String 1
tvntbjlrzmnanhjoarvwcfupzakhanoozzdnuguawwrcveurliswckbnwvzanajabvmevktdrjaowqhisljmkeabamndbjmhgrvttfptqtfeabhdlvlefewaoggkwusuplrfflgejwnwxhzfunsbzde
gafktfzuiotrbehwljayveammfnvaeauhatwhyqpaoomhsvubnwyavnoxyztsyavdavowuuvzwdarlqajalqtkbbkikrvlcfccmelrbqawtkmcgtbbtuaiksvnazaqqviakiitsvrzaoouamcaouisd
apelzrzbtwqpwhzgltisblaoouumimejmcehousblbrlnykhseziozhpmmltuztlilbfmstpbwgjiawudqnvtuwsfaylbuolwhtgwbttgumrqoozwfhlvo
Index of coincidence: 0.043903
m=3 String 2
ulzeghppbfqstywzalplgnmblalzylfoeewelutzrypesjhanaghzukqykdaruihyrzalgevzckgpiajhsmowazuiknlzojruvfwepwuuvzwhwavxvwipmownbwejybwzoeaadsqhcejnsicevaveaf
dkpwnjuilafnlagrvesglgtuzpbngyumnguuatkclwwfraamabymrarvaawbumrarjneeahwpnrzzdljybefulunsbaxvwnvzilqhkslfzvtlusgczvhweufivhvedwtnapmuwnalckapqhphczcjgs
onzrlpnqoumsfrlwewfavmlepkgkazgaefqcjcubrgnhxoznkmedsltpucpkcfuomezbzauvnieadtliedrvwgfgknmhykiamblbrwylgduvpwodfkzcgt
Index of coincidence: 0.043414
m=3 String 3
mplamacsrhelofjutlelqtdrflcabomowepmnguvdwsbualwzuzmgphavwrnlaqnpaekruwvsraprlfltafetbswzbjtwiaktlrjwceahwferjwgjvymnkaakrjekgtjrhiimommlvsluefbbkikejv
ejgyqomppadsgapewfywchuvccafsgtzqmmdwuhtslualaabvdeubjbrggzveubjdelptqtfepzsnjoiakxnwcevavhjvyvtgqoibsroliwulvokuhbnlnlmyqvwbefulgljyavhahpatfrlbesulqo
gtpbrsblmauvdbrkxntiohdstgezltsqppnlvadzfjnnhnabdkfsbmlyvmngrziatalkogplbuwasztljsmnauliigrbqeloaesgbjqrsfbbnlhhiolusg
Index of coincidence: 0.043175
m=4 String 1
tllbhczfenyjalecndzacalmzepuuuwysvjllazcuhwkrnuqbrevgwrcawifssfkasakjboagvrtpeqvfawwlvyfmaobjwytpoifdmjcsxsfuvizavapyfupofsegejswetvfbfeuzaudyktowasabn
yuvrrywvyrjanpuhfwrslliaenkuviajlntclirsowvucskbvniumnhwqwuapjtnhzkauhlazussgerrzqmqsdzwxiaoaetukleapecvsbfrhhkndzdbhpvlkrtotbbotvbgaswijnvawgiyhqoaatb
btlsmvnodihcs
Index of coincidence: 0.042321
m=4 String 2
uptgarbhatfoalwgtpllhyooeenlgarwcsarnuwzpnywaraayaeludzroplihamwbbibdzihultectuwehjdxvepkanrkjguzhfaoehvwnefekbejgkgtjmiadbapleyvgumpaaythgmhthawuhaabb
eaabxazsmbdjlwatznzrdojbxtlekbhvwvciolkrazwmuobzbaelvvvadfialiwvrcpoqrccsigopzbrnlwmvhlktfillsugzmgqcqluuzbnnyzbeesztympgzualzksulweaallqrnufiambuiohlg
byruuboohfzuo
Index of coincidence: 0.045557
m=4 String 3
mnemlprnsojztvlqubfkzboowdenuzdreuuazshgbqvzaljhpmartvsagrhjtjotauznlwmrtvwwpuhfwrhvjlinwwkgekswrramgqlnjuzcbsveedjknoularlawvfylhmzcvggunmwaupllmrauad
yrjoagtuuvreoeqvppazjaykqucbsvrvyfzqehsqfiklvtchuwnsiqzeevngiuavlhopfmheujqanpzpbtuuwrrlwtbmdokeizsmfnocdlgnnoasmfollmcnufiielmapbiwittddmtglfnrbklwbsw
wqgdbqwhwklvt
Index of coincidence: 0.041270
m=4 String 4
vzajpsmqlhwurplfmralanfozwmgtvwpbehwigmkkavdnainvzkkevjkpqallmeezwmntjjkrfjfwatzebagvwmeoagweubjleilsmwelhibnakdfefwqziptnghrwagcaucnnsamqtuwqcsoflvmvw
mbnvgzbeaadvetuwedznqjalfwbnakxvcvgmqbblltlgguthlkfyavbqtlkmysaaaatapbocldotllsboapfbgensvhopgmatjephjabrjlxnhkkismpumtczlmafzgpnujdzuesvwuskglyelmegr
jtgfrplzfolgg
Index of coincidence: 0.044404
m=5 String 1
tpejcbnlyotpctbacyooenntwweelnsmunvdnahveltvcorasaoesintohtftcutfhhgvenooreutzridelexecnieeeptoiodlheeyctmcnetntdtpswhamnernraterdeeutpdsdaaetcnihvctie
bsaitcoctnesyhaetalushcothcecsonlrntashreiimotgitacnjszglhzskdzlulgfltzmgvwwdwldvaffghulmtgwtsuqldflg
Index of coincidence: 0.055619
m=5 String 2
unahrrqnfzregurahbfzeegurrbjrzgcpqvrrjnrererrgqfhjeabznbirrrepaveralvieangeyurefoqwsnzbvbefagnzpargglfgeuznfyuquhucourvbbybrxgbybrvpaverrjjaxubvbrvncqq
rrftuutuvanfnvequaiynrhaurhougatrrboqvrgxfbheueaeqfhvuljhyakzslpvpuzollztlijsldsvulkybklalwjlubpohkls
```



```
Index of coincidence: 0.055524
m=5 String 3
mzbapmetjulwqmzlznmedmuadpvaaizzbakalibaakwzapiitmkbumjzmkvtwwqwwbwxlmmakwwgwliagmcwuiukvdvkkquianepvawgmcbagmamaqtwmlabdmvbazvmvdnwqwwzzqibqwukvxlvzmi
kqlvmvgbbwkmvzbwigmtvlaaqmbziqoebpbmmwbwttvaskmlgmpcudrrnohdeompmnctaefoubuaatenngsimbeahsrtrdrnowoco
Index of coincidence: 0.055809
m=5 String 4
vlglsfaowallfdlkalowwugzwssuwawgkwwaaqymkgdskwljlfwawkdwjglwfeufewdjwfkwgjjsjofmsjvjhfeskagjwfmltsawwsvhufagazgwwkalfsaawuaogwsuaaleufnanljkfkeskjwfgll
slwwlgkzuluvqvqfnkjwvakofpasjdgzzsqwufzkwsologkmseelcbfnnnnefshyctruiabspngatujrtugaryoobgbygfvohfhut
Index of coincidence: 0.059476
m=5 String 5
ltmpzhshjavlnpflaoozpluvycuhluhkhyznuapzvuvjaphlsmtzabljauvjpthzajvvypwabkkbphalmhnlsfbazjdfyjupfbarjylavpvsuhmuyhloaauvyajvyzuajjothzpzloylnlbaavyvcoh
bozklsbhhiliawdvlpiaazppalculspplzlupdllnaldpuzzjpqoabbnxkbmibtmmkzimbkapbeiziqmwwinlqiwebbqgmbwzizvg
Index of coincidence: 0.054765
m=6 String 1
tnblznnjavcuzkaozduuwrvulscbwznjbmvtrawhsjkaanbmgvtpqfahllfwogwsprfgjnxzuszeakfuorewjyemfveuawypomsunyvoytyvaouvwalaaqkbirlfcerqwkctbuisnzqvaitvzoumaus
aezztqwzlibaouiemeoslrnkszohmlutibmtbgiwdntwfybowtwtgmqowhv
Index of coincidence: 0.042789
m=6 String 2
uzgpbqtwapgmllyfewltrpshngzkydriyzlezkpahmwzinzjufewuzhaxwponwjbzeashenieaefkwjianareggupnymgutcwfambmavabmajeawnzdjbflnbxwviqklztugzhefvvdtamwacaqpccg
ozlnomflefvlpgageqjurnxzkestupcumzzunedlerwfkmyimlrygupofzg
Index of coincidence: 0.043108
m=6 String 3
mlmcreojteqdfcbmwpnudsulzzghvrlqperwsarftftszjwatrwehfrwjynakjktrimmlsufbievjyopasaefwhvcfgzmdutlaabdujrgvujepqfpsjikncvvjytqisoiuvkhnnmqweugjahhafleuq
gprbmudrxtodtelspnvdfnhadfblvnritlopbwstjmalirqlasbqsbnhils
Index of coincidence: 0.039371
m=6 String 4
vtjrmahorwfpahnozngawceriwknvaaavekdjoqilmebmdjhrtfttebdveeagkuulflewwhfnbdgftzitbhlavamnaahthqaohvbwanxzsadvwuzdrqjltbkkvccmlbatmgbtakvaaqikisraoacoid
plrbwphgtsloummjchubblyheizpmtzllfspwjauqvusalulhgbturozflo
Index of coincidence: 0.044065
m=6 String 5
lehpfsyzllnbazloeeuzyejaahuqkauhragvcgijsoauklorvwpuvwwvvimwbeywoadqcjscvvadpnulflgvsltzbgunuaklwraayrrawurrnehprzlyeuusavnzlhsfvlscvwuihewnpunlkphhzjs
nrpqusrwwamekkzafccbghonmdlpckfoebaviatidvggnhkabbwldvwdkct
Index of coincidence: 0.044247
m=6 String 6
paashlfulltrlaooemgvwbawumpawnanakuvrpllaebwbtikljcawejgvmkaregjhiomvlebkkjegqmpdgpwycucastqmwhsulavebbgzebdltteznoaxweahvvgobrlwloubllyvbfllyvaptrbslo
tbslavbknihsgztqplazjnnbksmymgzaakgluazlsnuigbeoegjrfblhoug
Index of coincidence: 0.048941


Enter correct length of keyword to continue : 5

Table:
g         Vg1             Vg2             Vg3             Vg4             Vg5
0         0.0592          0.0434          0.0385          0.0391          0.0362
1         0.0388          0.0383          0.0344          0.0391          0.0385
2         0.0371          0.0391          0.0352          0.0400          0.0329
3         0.0335          0.0423          0.0362          0.0428          0.0385
4         0.0396          0.0367          0.0400          0.0355          0.0354
5         0.0340          0.0312          0.0326          0.0413          0.0337
6         0.0370          0.0388          0.0356          0.0389          0.0363
7         0.0392          0.0418          0.0413          0.0451          0.0585
8         0.0343          0.0341          0.0589          0.0360          0.0473
9         0.0358          0.0417          0.0401          0.0370          0.0359
10        0.0357          0.0337          0.0336          0.0326          0.0300
11        0.0479          0.0340          0.0324          0.0358          0.0415
12        0.0375          0.0392          0.0435          0.0367          0.0362
13        0.0399          0.0593          0.0364          0.0391          0.0375
14        0.0385          0.0371          0.0368          0.0444          0.0357
15        0.0468          0.0308          0.0365          0.0333          0.0353
16        0.0364          0.0390          0.0322          0.0328          0.0345
17        0.0352          0.0450          0.0345          0.0374          0.0364
18        0.0358          0.0347          0.0439          0.0615          0.0419
19        0.0393          0.0358          0.0457          0.0376          0.0400
20        0.0365          0.0369          0.0361          0.0339          0.0423
21        0.0354          0.0322          0.0402          0.0294          0.0407
22        0.0397          0.0370          0.0436          0.0467          0.0459
23        0.0326          0.0407          0.0393          0.0341          0.0418
24        0.0348          0.0415          0.0361          0.0365          0.0356
25        0.0404          0.0366          0.0374          0.0343          0.0326

Keyword:
anish

The plain text is: thedepartmentofjusticehasbeenandwillalwaysbecommittedtoprotectingthelibertyandsecurityofthosewhomweserveinrecentmonthshoweverwehaveo
nanewscaleseenmainstreamproductsandservicesdesignedinawaythatgivesuserssolecontroloveraccesstotheirdataasaresultlawenforcementissometimesunabletorecove
rthecontentofelectroniccommunicationsfromthetechnologyprovidereveninresponsetoacourtorderordulyauthorizedwarrantissuedbyafederaljudgeforexamplemanycomm
unicationsservicesnowencryptcertaincommunicationsbydefaultwiththekeynecessarytodecryptthecommunicationssolelyinthehandsoftheenduserthisappliesbothwhent
hedataisinmotionoverelectronicnetworksoratrestonanelectronicdeviceifthecommunicationsproviderisservedwithawarrantseekingthosecommunicationstheproviderc
annotprovidethedatabecauseithasdesignedthetechnologysuchthatitcannotbeaccessedbyanythirdpartywedonothaveanysilverbulletsandthediscussionswithintheexecu
tivebrancharestillongoingwhiletherehasnotyetbeenadecisionwhethertoseeklegislationwemustworkwithcongressindustryacademicshmjnuuthjimktshvjuzyjnugwjvgluf
vqhlgvdznzvusxvmfkmwnbdfgauzyepmlchgfuiekflcfbdghuzsfmlcblbsqftywiuzyxjdmmgatggmxivytvuwvmoxwwsibdfsbswylcblqwrjdffzfvifbpahycpfykobfxaiggle(base) anis
h@Anishs-MacBook-Pro Lab (base) anish@Anishs-MacBook(base) anish@Anish(base) an(base) an(base) an(base) an(base) an(base) an(base) an(base) an(base) an
(base) anish@Anishs-MacBook-Pro Lab %
```

Questions:

1.What are various ways of cryptanalysis? Explain any two methods.

-> Cryptanalysis of the Vigenere Cipher is actually a two step process that involves finding the length of the keyword first and then figuring out the keyword itself.

Method 1: Displacement-Coincidences

This method involves comparing the cipher text against itself with shifted places. The simple nature of this method makes it easy to implement even on a piece of paper.

As aforementioned, you begin by fixing the position one cipher text’s letters and then compare the another string/paper with the san cipher text by shifting the places one by one until all places are exhausted.

Whilst shifting places, the cryptanalyst must take of the number of times a particular position has the same letter on either text.

Finally, the results are to be tabulated and the shift with the most number of coincidences is guessed/assumed to be the keyword's length.

Once the keyword length has been figured out, it's just a matter of performing a frequency analysis on the sets of letters of the cipher that each position of the keyword corresponds to.

Method 2: Friedman Test

This is the most mathematicaly complex method for the cryptanalysis of the Vigenere cipher.

The first step is to guess the length of the keyword, assuming m=1,2,3,4…

Depending on the value of m, m sub strings are to be created taking every m^{th} letter of the string for the sub string, shifting the starting placer by a unit for each sub string.

After creating the substrings we find the Index of Coincidence (explained in question 3) of each sub string and if our assumed value of m has substrings whose Index of Coincidence has values near 0.065, we can be fairly certain that our assumption is valid.

Once the keyword length has been determined, we must move on to finding the actual keyword. For this, we firstly need to keep a frequency table ready that charts the frequencies of the letters of the particular language's alphabet (Eg: The table from Cryptographical Mathematics by Robert Edward Lewand for the English alphabet) in usual text.

For each sub string for the particular value of m, we are to create a vector q that would consist of the frequency of appearance of each character of the alphabet divided by the length of the substring.

Following the creation of m such vectors, we are to create another vector v_g that would consist of the elements of q shifted by g places, with g=0-25. As such, for every vector q, we would have 26 v_g vectors.

After the creation of v_g vectors, we proceed to compute the dot product of p and every v_g. This would result in a new vector M for every q that would consist of 26 values. These values are to be arranged in respective columns and from each column, the highest value (which would be near 0.065) is to be chosen. Corresponding to the values, their respective shift (g) is to be noted and using the alphabet encoding table where every letter from A-Z is assigned a numeric value from 0-25, we find out the corresponding letters and compose them together to form the keyword.

After this point, we simply decrypt the text using the keyword's alphabetically assigned number to subtract them from the cipher text's numeric values and cycling the keyword until the ciphertext is exhausted.

2.Explain Kasiski's method with an example.

Kasiski's method is the third method of cryptanalysis of the Vigenere cipher which is concerned with the presence of digrams, trigrams and so on, in the cipher text.

Therefore, we create a frequency table of those digrams and trigrams appearing more than once and their position occurrence.

We then take two of those groups of letter that have highest frequency and compute the difference of the starting positions of their consecutive occurrences. We then take the gcd of the position differences of the two groups which would work out to be the keyword length.

Once the keyword length has been figured out, it’s just a matter of performing a frequency analysis on the sets of letters of the cipher that each position of the keyword corresponds to.

Eg:krgclgcgsbvcxxjimldlcwmbnpcyjqdybimlqjmblgcivkqqkrbrigclyfmlqpmdwmpjsxhmsresx

n-grams -> frequency -> starting posns. -> difference

iml: 2 ->16, 34 ->18

lgc: 2 ->5, 41 -> 36

gcl: 2 ->3, 54 -> 51

mlq: 2 ->35, 59 -> 24

gcd(18,36)=18 (unlikely since the number is quite high, reserve it for later test)

gcd(18,51)=3

Consider keyword length=3,

Using tool: https://crypto.interactive-maths.com/kasiski-analysis-breaking-the-code.html#act

We decipher the keyword to be KEY and the plain text to be:

anishiscurrentlyinthemiddleofstudyingforhisexamsandheishavinglotsoffundoingit

3.Explain Index of Coincidence.

The index of coincidence for a string or particular group of letters is the sum of the frequency of every letter’s appearance in the string (assume equal to n) multiplied by (n-1) and the whole sum divided by the product of the length of the string times minus 1 of the length.

Eg; Assume string x= anishstudiescryptography

It consists of the letter frequency

A	2
B	0
C	1
D	1
E	1
F	0
G	1
H	2
I	2
J	0
K	0
L	0
M	0
N	1
O	1

P	2
Q	0
R	2
S	3
T	2
U	1
V	0
W	0
X	0
Y	2
Z	0

And the length of x = 24

So, the $I_c(x) = ((2)(1) + (2)(1) + (2)(1) + (2)(1) + (2)(1) + (3)(2) + (2)(1) + (2)(1))/((24)(23))$

[We ignore the letters with frequency 1 since their product would turn out to be 0]

Therefore, the $I_c(x)=0.036231$