```java
/*
 * Kai Zhao
 * MCS 425
 * Final Project
 * 2014 April 28
 *
 * This java program takes an CIPHERTEXT input, and attempts to decrypt it using a dictionary.
 * This program currently works for Ceasar Ciphers, Additive Ciphers, Multiplicative Ciphers, Affine Ciphers, and Vigenère (need repeating sequences) Ciphers
 * Decrypting Ceasar Ciphers will try all 26 combinations and print the plain text with most matches in the dictionary
 * Decrypting Additive/Multiplicative/Affine Ciphers will try multiplicative keys that are relatively prime to 26, then shift, and print the plain text with most matches in the dictionary
 * Decrypting Vigenère Ciphers will use Kasiski's Test to find the length of the keyword, then try 26^(length of keyword) for the largest repeating sequence in the CIPHERTEXT, try to form a word in the dictionary, and then print the plain text with most matches in the dictionary
 */

import java.io.File;


public class MCS425DecryptClassicalCiphers {

    Scanner keyboard = new Scanner( System.in);     // used to read user input
    ArrayList<String> dictionary = new ArrayList<String>();

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        MCS425DecryptClassicalCiphers instanceOfMCS425DecryptClassicalCiphers = new MCS425DecryptClassicalCiphers();
        instanceOfMCS425DecryptClassicalCiphers.start();
    }

    public void start() throws IOException {
        readInDictionaryWords();        // read dictionary
        while (true) {
            System.out.print("\n> ");
            String s = keyboard.nextLine().toUpperCase();
            if (s.length() == 0) {
                continue;
            }
            if (s.length() - s.replace(" ", "").length() + 1 >=  0.5*s.replace(" ", "").length()) {
                int startingIndex = s.contains("26") ? 1 : 0;
                {                       // formatting
                    for (int i = 0; i < 26; i++) {
                        s = s.replace(String.format("%02d", i + startingIndex), ""+ (char) ('A'+i));
                    }
                }                       // formatting
                for (int i = 6; i >=2; i--) {
                    String spaces = "";
                    while (spaces.length() < i) {
                        spaces += " ";
                    }
                    s = s.replace(spaces, "<space>");
```

```
52                }                          // formatting
53                s = s.replace("       ", "<space>").replace("      ", "<space>").replace("
  ", "<space>").replace("    ", "<space>").replace("   ", "<space>").replace(" ",
  "").replace("<space>", " ");
54            }
55            int maxCeasarCipher = 0;
56            int maxAffineCipher = 0;
57            int maxNumberOfWordsMatched = 0;
58            ArrayList<String> plaintext = new ArrayList<String>();
59            // ** Caesar Cipher
  ****************************************************************
60            maxNumberOfWordsMatched = 0;
61            plaintext = new ArrayList<String>();
62            ArrayList<Integer> CaesarKeys = new ArrayList<Integer>();
63            for (int CaesarKey = 0; CaesarKey < 26; CaesarKey++) {
64                int numberOfWordsMatched = 0;
65                char[] c = s.toCharArray();
66                for (int charIndex = 0; charIndex < c.length; charIndex++) {
67                    if (c[charIndex] < 'A' || c[charIndex] > 'Z') {
68                        continue;
69                    }                      // skip non letter characters
70                    c[charIndex] = (char) (c[charIndex] - CaesarKey < 'A' ? c[charIndex] -
  CaesarKey + 26 : c[charIndex] - CaesarKey);
71                }
72                String[] words = new String(c).split(" ");
73                boolean allWordsInDictionary = true;
74                for (String word : words) { // formatting
75                    if (dictionary.contains(word.replace(".", "").replace(",",
  "").replace("!", "").replace("-", "").replace(";", ""))) {
76                        numberOfWordsMatched++;
77                    } else {
78                        allWordsInDictionary = false;
79                    }
80                }
81                if (allWordsInDictionary) { // print plaintext f all words are in the
  dictionary
82                    System.out.print("CaesarKey = " + CaesarKey + ": ");
83                    System.out.println(new String(c));
84                }
85                if (numberOfWordsMatched == maxNumberOfWordsMatched) {
86                    plaintext.add(new String(c));
87                    CaesarKeys.add(CaesarKey);
88                }
89                if (numberOfWordsMatched > maxNumberOfWordsMatched) {
90                    maxNumberOfWordsMatched = numberOfWordsMatched;
91                    plaintext = new ArrayList<String>();
92                    plaintext.add(new String(c));
93                    CaesarKeys = new ArrayList<Integer>();
94                    CaesarKeys.add(CaesarKey);
95                }
96            }                              // else print the message that has the mot words in
  the dictionary
97            if (maxNumberOfWordsMatched > 0 && maxNumberOfWordsMatched < s.split(" ").length)
  {
98                for (int key = 0; key < plaintext.size(); key++) { //String p : plaintext) {
99                    System.out.print("CaesarKey = " + CaesarKeys.get(key) + ": ");
100                   System.out.println(plaintext.get(key));
```

```
101                 }
102             }
103             maxCeasarCipher = maxNumberOfWordsMatched;
104             // ** End Caesar Cipher ************************************************************
105             // ** Affine/Multiplicative Cipher ************************************************
106             maxNumberOfWordsMatched = 0;
107             plaintext = new ArrayList<String>();
108             int[] MultiplicativeKeys = {3, 5, 7, 9, 11, 13, 17, 19, 21, 23};
109             ArrayList<Integer> MultKeys = new ArrayList<Integer>();
110             ArrayList<Integer> AddKeys = new ArrayList<Integer>();
111             for (int MultiplicativeKey = 0; MultiplicativeKey < MultiplicativeKeys.length; MultiplicativeKey++) {
112                 for (int AdditiveKey = 0; AdditiveKey < 26; AdditiveKey++) {
113                     int numberOfWordsMatched = 0;
114                     char[] c = s.toCharArray();
115                     for (int charIndex = 0; charIndex < c.length; charIndex++) {
116                         if (c[charIndex] < 'A' || c[charIndex] > 'Z') {
117                             continue;
118                         }               // skip non alpha characters
119                         int newChar = c[charIndex] * MultiplicativeKeys[MultiplicativeKey] + AdditiveKey;
120                         while (newChar < 'A') {
121                             newChar += 26;
122                         }               // do Vigenère computation
123                         while (newChar > 'Z') {
124                             newChar -= 26;
125                         }
126                         c[charIndex] = (char) newChar;
127                     }
128                     String[] words = new String(c).split(" ");
129                     boolean allWordsInDictionary = true;
130                     for (String word : words) { // formatting
131                         if (dictionary.contains(word.replace(".", "").replace(",", "").replace("!", "").replace("-", "").replace(";", ""))) {
132                             numberOfWordsMatched++;
133                         } else {
134                             allWordsInDictionary = false;
135                         }
136                     }                   // find the keys
137                     int multiInverse = MultiInverse(MultiplicativeKeys[MultiplicativeKey]);
138                     int addInverse = AddInverse(c[0], s.charAt(0), multiInverse);
139                     if (allWordsInDictionary) { // print out plaintext if all words are in the dictionary
140                         System.out.print("MultiplicativeKey = " + multiInverse + ", Additive Key = " + addInverse + ": ");
141                         System.out.println(new String(c));
142                     }
143                     if (numberOfWordsMatched == maxNumberOfWordsMatched) {
144                         plaintext.add(new String(c));
145                         MultKeys.add(multiInverse);
146                         AddKeys.add(addInverse);
147                     }                   // keep track of the plaintext with most words in the dictionary
148                     if (numberOfWordsMatched > maxNumberOfWordsMatched) {
149                         maxNumberOfWordsMatched = numberOfWordsMatched;
```

```
150                         plaintext = new ArrayList<String>();
151                         plaintext.add(new String(c));
152                         AddKeys = new ArrayList<Integer>();
153                         MultKeys = new ArrayList<Integer>();
154                         MultKeys.add(multiInverse);
155                         AddKeys.add(addInverse);
156                     }
157                 }
158             }                           // print the plaintext with most words in the dictionary
159             if (maxNumberOfWordsMatched > 0 && maxNumberOfWordsMatched < s.split(" ").length && maxCeasarCipher < s.split(" ").length) {
160                 for (int key = 0; key < plaintext.size(); key++) {
161                     System.out.print("MultiplicativeKey = " + MultKeys.get(key) + ", Additive Key = " + AddKeys.get(key) + ": ");
162                     System.out.println(plaintext.get(key));
163                 }
164             }
165             maxAffineCipher = maxNumberOfWordsMatched;
166             // ** End Affine/Multiplicative Cipher ***********************************************
167             if (maxCeasarCipher + 1 >= s.split(" ").length || maxAffineCipher + 1 >= s.split(" ").length) {
168                 continue;               // check if affine cipher is required
169             }
170             // ** Vigenère Cipher ****************************************************************
171             maxNumberOfWordsMatched = 0;// run Kasiski's Test
172             plaintext = new ArrayList<String>();
173             int[] possibleKeyLengths = new int[21];
174             String s2 = s.replace(" ", "").replace(".", "").replace(",", "").replace("!", "").replace("-", "").replace(";", "");
175             ArrayList<String> sequences = new ArrayList<String>();
176             ArrayList<String> repeatingSequences = new ArrayList<String>();
177             for (int repeatLength = 12; repeatLength >= 3; repeatLength--) {
178                 for (int charPosition = 0; charPosition <= s2.length() - repeatLength; charPosition++) {
179                     String s3 = s2.substring(charPosition, charPosition+repeatLength);
180                     if (sequences.contains(s3)) {
181                         boolean notAlreadyIncluded = true;
182                         for (String s4 : repeatingSequences) {
183                             if (s4.contains(s3)) {
184                                 notAlreadyIncluded = false;
185                                 break;
186                             }
187                         }
188                         if (notAlreadyIncluded) {
189                             repeatingSequences.add(s3);
190                         }
191                     } else {
192                         sequences.add(s3);
193                     }
194                 }
195             }                           // compare sequences of letters
196             for (int sequence = 0; sequence < repeatingSequences.size(); sequence++) {
197                 ArrayList<Integer> occurances = new ArrayList<Integer>();
198                 occurances.add(s2.indexOf(repeatingSequences.get(sequence)));
```

```
199                 while (s2.indexOf(repeatingSequences.get(sequence),
    occurances.get(occurances.size() - 1) + 1) > 0) {
200                     occurances.add(s2.indexOf(repeatingSequences.get(sequence),
    occurances.get(occurances.size() - 1) + 1));
201                 }                       // find the difference between repeating sequences
202                 for (int occurance = 0; occurance < occurances.size(); occurance++) {
203                     for (int occurance2 = occurance + 1; occurance2 < occurances.size();
    occurance2++) {
204                         int distanceBetweenPositions = occurances.get(occurance2) -
    occurances.get(occurance);
205                         for (int possibleKeyLength = 1; possibleKeyLength <
    possibleKeyLengths.length; possibleKeyLength++) {
206                             if (distanceBetweenPositions % possibleKeyLength == 0) {
207                                 possibleKeyLengths[possibleKeyLength]++;
208                             }
209                         }
210                     }
211                 }
212             }
213             int maxKeyLength = 1;
214             for (int possibleKeyLength = 2; possibleKeyLength < possibleKeyLengths.length;
    possibleKeyLength++) {
215                 if (possibleKeyLengths[possibleKeyLength] >= possibleKeyLengths[maxKeyLength])
    {
216                     maxKeyLength = possibleKeyLength;
217                 }
218             }
219             if (maxKeyLength >= 10) {
220                 continue;
221             }
222             if (maxNumberOfWordsMatched < s.split(" ").length && maxCeasarCipher < s.split("
    ").length && maxAffineCipher < s.split(" ").length) {
223                 System.out.print("VigenèreKeyLength = " + maxKeyLength);
224             }
225             int[] VigenèreKeys = new int[maxKeyLength];
226             for (int VigenèreGroup = 0; VigenèreGroup < Math.pow(26, maxKeyLength);
    VigenèreGroup++) {
227                 char[] c = repeatingSequences.get(0).toCharArray(); // start fresh
228                 for (int charIndex = 0; charIndex < repeatingSequences.get(0).length();
    charIndex++) {
229                     c[charIndex] = (char) (c[charIndex] - VigenèreKeys[charIndex %
    maxKeyLength] < 'A' ? c[charIndex] - VigenèreKeys[charIndex % maxKeyLength] + 26 :
    c[charIndex] - VigenèreKeys[charIndex % maxKeyLength]);
230                 }                       // shift every letter according to VigenèreKeys
231                 if (dictionary.contains(new String(c))) { // check if the repeating sequence
    matches the dictionary
232                     char[] keyword = new char[maxKeyLength];
233                     for (int k = 0; k <maxKeyLength; k++) {
234                         keyword[k] = (char) ('A' + VigenèreKeys[(k +
    s2.indexOf(repeatingSequences.get(0))) % maxKeyLength]);
235                     }
236                     System.out.print(", VigenèreKey = " + new String(keyword) + ": ");
237                     int counter = 0;
238                     c = s.toCharArray();
239                     for (int charIndex = 0; charIndex < c.length; charIndex++) {
240                         if (c[charIndex] < 'A' || c[charIndex] > 'Z') {
241                             continue;   // ignore non alpha characters
```

```
242                         }
243                         int newChar = c[charIndex] - (keyword[counter++%maxKeyLength] - 'A');
244                         if (newChar < 'A') {
245                             newChar += 26;
246                         }
247                         c[charIndex] = (char) newChar;
248                     }
249                     System.out.println(new String(c));
250                     break;
251                 }
252                 VigenèreKeys[maxKeyLength - 1]++; // update the next key to try
253                 for (int VigenèrePosition = maxKeyLength - 1; VigenèrePosition > 0;
    VigenèrePosition--) {
254                     if (VigenèreKeys[VigenèrePosition] >= 26) {
255                         VigenèreKeys[VigenèrePosition] -= 26;
256                         VigenèreKeys[VigenèrePosition - 1]++;
257                     }
258                 }
259             }
260             // ** End Vigenère Cipher
    ************************************************************
261         }
262     }
263
264     public int MultiInverse(int key) {
265         int[] MultiplicativeKeys = {3, 5, 7, 9, 11, 13, 17, 19, 21, 23};
266         for (int ky = 0; ky < MultiplicativeKeys.length; ky++) {
267             if ((MultiplicativeKeys[ky] * key) % 26 == 1) {
268                 return MultiplicativeKeys[ky];
269             }
270         }
271         return 0;
272     }
273
274     public int AddInverse(char a, char b, int multKey) {
275         int returnValue = (b - 'A') - (((a - 'A')*multKey)%26);
276         if (returnValue < 0) {
277             returnValue += 26;
278         }
279         return returnValue;
280     }
281
282     public void readInDictionaryWords() throws IOException
283     {
284         // Define a Scanner to read from an input file.  Note that the name of
285         // the file given in the code below MUST match the actual filename of
286         // the dictionary file.  This file should be in the same directory
287         // as the source code for WordCross.java
288         File dictionaryFile = new File("dictionary.txt");    // declare the file
289         // ensure file exists and is in the correct directory
290         if( ! dictionaryFile.exists()) {
291             // print the directory where this program expects to find dictionary
292             System.out.println("Your dictionary file should be placed in " +
    System.getProperty("user.dir"));
293             System.out.println("*** Error *** \n" +
294                     "Your dictionary file has the wrong name or is " +
295                     "in the wrong directory.  \n" +
```

```
296                     "Aborting program...\n\n");
297             System.exit( -1);    // Terminate the program
298         }
299         Scanner inputFile = new Scanner( dictionaryFile);
300         // while there are words in the input file, add them to the dictionary
301         while( inputFile.hasNext()) {
302             dictionary.add( inputFile.nextLine().toUpperCase() );
303         }
304         inputFile.close();
305     }//end createDictionary()
306 }
307
```

Program Sample Input/Output

Every line that starts with “>” is the input. The lines of text that follows are the output.

```
> B CVNCMJOH CFF
CaesarKey = 1: A BUMBLING BEE

> WKH FDHVDU FLSKHU LV QDPHG DIWHU MXOLXV FDHVDU ZKR XVHG LW ZLWK D VKLIW RI WKUHH
CaesarKey = 3: THE CAESAR CIPHER IS NAMED AFTER JULIUS CAESAR WHO USED IT WITH A
SHIFT OF THREE

> QLAP ATTKSPKB EK EWGP FXWTWYNBVI QAG PLK XKAVURAPUWN PLAP PLK GSUKNSKG WT
SXIFPWCXAFLI ANB EAPLKEAPUSG AXK HKXI KVKCANP, FYXK GSUKNSKG. U TWYNB PLAP PLK KNBG
TWX QLUSL PLKGK FYXK AXK YGKB AXK VKGG KVKCANP.
CaesarKey = 23: TODS DWWNVSNE HN HZJS IAZWZBQEYL TDJ SON ANDYXUDSXZQ SODS SON
JVXNQVNJ ZW VALISZFADIOL DQE HDSONHDSXVJ DAN KNAL NYNFDQS, IBAN JVXNQVNJ. X WZBQE
SODS SON NQEJ WZA TOXVO SONJN IBAN DAN BJNE DAN YNJJ NYNFDQS.
MultiplicativeKey = 9, Additive Key = 0: WHAT AFFECTED ME MOST PROFOUNDLY WAS THE
REALIZATION THAT THE SCIENCES OF CRYPTOGRAPHY AND MATHEMATICS ARE VERY ELEGANT, PURE
SCIENCES. I FOUND THAT THE ENDS FOR WHICH THESE PURE ARE USED ARE LESS ELEGANT.

> 25 22    15 05 11    13 17 04 10    10 05    18 21    17    19 08 15 06 10 05 23 08
17 06 24 21 08 ,    09 10 11 20 15    03 17 10 24 21 03 17 10 25 19 09 .    - -    18
08 11 19 21    09 19 24 04 21 25 21 08
CaesarKey = 17: IF YOU WANT TO BE A CRYPTOGRAPHER, STUDY MATHEMATICS. -- BRUCE
SCHNEIER
MultiplicativeKey = 0, Additive Key = 25: AN AAA AANN NA NA A ANANNAANANNAN, ANANA
AANNAAANAAA. -- NNAAA AANNAAAN
VigenèreKeyLength = 5, VigenèreKey = MAAAA: NW PFL BREK KT SV R TWPGKFLIRGYJI, JKLIP
DRKMVDRKNTJ. -- SIZTV JTMEVZVW

> WKH FDHVDU FLSKHU LV QDPHG DIWHU MXOLXV FDHVDU ZKR XVHG LW ZLWK D VKLIW RI WKUHH
CaesarKey = 3: THE CAESAR CIPHER IS NAMED AFTER JULIUS CAESAR WHO USED IT WITH A
SHIFT OF THREE

> QLAP ATTKSPKB EK EWGP FXWTWYNBVI QAG PLK XKAVURAPUWN PLAP PLK GSUKNSKG WT
SXIFPWCXAFLI ANB EAPLKEAPUSG AXK HKXI KVKCANP, FYXK GSUKNSKG. U TWYNB PLAP PLK KNBG
TWX QLUSL PLKGK FYXK AXK YGKB AXK VKGG KVKCANP.
CaesarKey = 23: TODS DWWNVSNE HN HZJS IAZWZBQEYL TDJ SON ANDYXUDSXZQ SODS SON
JVXNQVNJ ZW VALISZFADIOL DQE HDSONHDSXVJ DAN KNAL NYNFDQS, IBAN JVXNQVNJ. X WZBQE
SODS SON NQEJ WZA TOXVO SONJN IBAN DAN BJNE DAN YNJJ NYNFDQS.
MultiplicativeKey = 9, Additive Key = 0: WHAT AFFECTED ME MOST PROFOUNDLY WAS THE
REALIZATION THAT THE SCIENCES OF CRYPTOGRAPHY AND MATHEMATICS ARE VERY ELEGANT, PURE
SCIENCES. I FOUND THAT THE ENDS FOR WHICH THESE PURE ARE USED ARE LESS ELEGANT.

> 25 22    15 05 11    13 17 04 10    10 05    18 21    17    19 08 15 06 10 05 23 08
17 06 24 21 08 ,    09 10 11 20 15    03 17 10 24 21 03 17 10 25 19 09 .    - -    18
08 11 19 21    09 19 24 04 21 25 21 08
CaesarKey = 17: IF YOU WANT TO BE A CRYPTOGRAPHER, STUDY MATHEMATICS. -- BRUCE
SCHNEIER
MultiplicativeKey = 0, Additive Key = 25: AN AAA AANN NA NA A ANANNAANANNAN, ANANA
AANNAAANAAA. -- NNAAA AANNAAAN
VigenèreKeyLength = 5, VigenèreKey = MAAAA: NW PFL BREK KT SV R TWPGKFLIRGYJI, JKLIP
DRKMVDRKNTJ. -- SIZTV JTMEVZVW
```

> XZD DTB BUIBKZEBT GZO EZTYHK PHVK. XZD DTB VY GZO KZIB KBYYBOT, MDTVUBTT KBYYBOT HUW XZDO PBWVRHK OBRZOWT. YQB ZUKX NHX YZ ROBHYB YQB WVLVYHK BJDVIHKBUY ZG HU BUIBKZEB VT YZ BUROXEY. -- EQVK CVPPBOPHU
CaesarKey = 21: CEI IYG GZNGPEJGY LET JEYDMP UMAP. CEI IYG AD LET PENG PGDDGTY, RIYAZGYY PGDDGTY MZB CEIT UGBAWMP TGWETBY. DVG EZPC SMC DE WTGMDG DVG BAQADMP GOIANMPGZD EL MZ GZNGPEJG AY DE GZWTCJD. -- JVAP HAUUGTUMZ
MultiplicativeKey = 5, Additive Key = 7: YOU USE ENVELOPES FOR POSTAL MAIL. YOU USE IT FOR LOVE LETTERS, BUSINESS LETTERS AND YOUR MEDICAL RECORDS. THE ONLY WAY TO CREATE THE DIGITAL EQUIVALENT OF AN ENVELOPE IS TO ENCRYPT. -- PHIL ZIMMERMAN

> AMGXTZ SVTYHICKVIH KNR WMYT NWH GGIPO P AVQETR WJJFXXBHXXWA GXXUIG. JHX IPR ZXORRTZR GXXUIG UNWZA SVTYHICKVIH, UNOXVT MI PNVS BB GGIPO. LM HWTL N WWWEX ZMLADZQ XD MAGGGCX. JAR E AWAKTZ BRT BB QPSR E BMFWPOR LPZQIG BB GGIPO.
CaesarKey = 15: LXRIEK DGEJSTNVGTS VYC HXJE YHS RRTAZ A LGBPEC HUUQIIMSIIHL RIIFTR. USI TAC KIZCCEKC RIIFTR FYHKL DGEJSTNVGTS, FYZIGE XT AYGD MM RRTAZ. WX SHEW Y HHHPI KXWLOKB IO XLRRRNI. ULC P LHLVEK MCE MM BADC P MXQHAZC WAKBTR MM RRTAZ.
MultiplicativeKey = 0, Additive Key = 0: NNNAAA NAANANNNANA NAA NNNA ANA NNNAN A NANNAA NAAAAAAAAANN NAANNN. AAA NAA AANAAAAA NAANNN NANAN NAANANNNANA, NANAAA NN AAAN AA NNNAN. AN ANAA A NNNNA ANANAAN AA NNNNNNA. ANA N NNNNAA AAA AA NANA N ANANANA AAANNN AA NNNAN.
VigenèreKeyLength = 4, VigenèreKey = PINE: LETTER FREQUENCIES CAN HELP YOU CRACK A SIMPLE SUBSTITUTION CIPHER. BUT THE VIGENERE CIPHER MASKS FREQUENCIES, MAKING IT HARD TO CRACK. WE USED A SHORT KEYWORD TO ENCRYPT. USE A LONGER ONE TO MAKE A MESSAGE HARDER TO CRACK.

> HXAE AE C CRRAPO OPIBWVHOL MOEECUO.
CaesarKey = 0: HXAE AE C CRRAPO OPIBWVHOL MOEECUO.
CaesarKey = 2: FVYC YC A APPYNM MNGZUTFMJ KMCCASM.
CaesarKey = 12: VLOS OS Q QFFODC CDWPKJVCZ ACSSQIC.
CaesarKey = 20: NDGK GK I IXXGVU UVOHCBNUR SUKKIAU.
MultiplicativeKey = 3, Additive Key = 2: THIS IS A AFFINE ENCRYPTED MESSAGE.

What did the skeleton ask for when he went into the coffee shop?
> O QID CT QCTTSS OBR O ACD
CaesarKey = 14: A CUP OF COFFEE AND A MOP

What kind of number transforms?
> WXBQUCA XZQUM!
CaesarKey = 8: OPTIMUS PRIME!

What is the scariest side of a haunted house?
> YMJ NSXNIJ!
CaesarKey = 5: THE INSIDE!

What do seconds, minutes, and months have that hours, days, and years do not have?
> 22 10 07 14 07 22 22 07 20 16
CaesarKey = 3: THE LETTER N
MultiplicativeKey = 5, Additive Key = 16: WET KTWWTG A

How do you get a kleenex to dance?
> KPO V GDOOGZ WJJBT DIOJ DO
CaesarKey = 21: PUT A LITTLE BOOGY INTO IT

What is a prehistoric monster called when it sleeps?
> Z CHMNRMNQD!

CaesarKey = 17: I LQVWAVWZM!
CaesarKey = 25: A DINOSNORE!
MultiplicativeKey = 9, Additive Key = 25: A JYNQCNQZM!
MultiplicativeKey = 9, Additive Key = 5: I RGVYKVYHU!
MultiplicativeKey = 21, Additive Key = 25: A PONSMNSHU!
MultiplicativeKey = 21, Additive Key = 13: I XWVAUVAPC!
MultiplicativeKey = 0, Additive Key = 25: A VENUWNUPC!
MultiplicativeKey = 0, Additive Key = 25: I DMVCEVCXK!
MultiplicativeKey = 3, Additive Key = 25: A BUNWGNWXK!
MultiplicativeKey = 3, Additive Key = 1: I JCVEOVEFS!
MultiplicativeKey = 19, Additive Key = 25: A HKNYQNYFS!
MultiplicativeKey = 19, Additive Key = 3: I PSVGYVGNA!
MultiplicativeKey = 0, Additive Key = 25: A NANAANANA!
MultiplicativeKey = 0, Additive Key = 25: I VIVIIVIVI!
MultiplicativeKey = 23, Additive Key = 25: A ZGNEUNEDQ!
MultiplicativeKey = 23, Additive Key = 23: I HOVMCVMLY!
MultiplicativeKey = 11, Additive Key = 15: I NEVOMVOTG!
MultiplicativeKey = 11, Additive Key = 25: A FWNGENGLY!
MultiplicativeKey = 5, Additive Key = 11: I TUVQWVQBO!
MultiplicativeKey = 5, Additive Key = 25: A LMNIONITG!
MultiplicativeKey = 17, Additive Key = 19: I ZKVSGVSJW!
MultiplicativeKey = 17, Additive Key = 25: A RCNKYNKBO!

When is a door not a door?
> DOLU PA'Z HQHY!
CaesarKey = 7: WHEN IT'S AJAR!

What do cannibals do at a wedding?
> 22 10 07 01 22 17 03 21 22 22 10 07 04 20 11 06 07 03 16 06 09 20 17 17 15
CaesarKey = 3: THEY TOAST THE BRIDE AND GROOM

Why did Tigger look inside the toilet?
> YV NRJ CFFBZEX WFI GFFY!
CaesarKey = 17: HE WAS LOOKING FOR POOH!

Which is bigger: Mrs. Bigger or Mrs. Bigger's baby?
> 05 19 16 13 12 13 10 20 04 21 06 04 05 12 23 20 05 05 23 16 13 20 18 18 16 03
CaesarKey = 12: THE BABY IS JUST A LITTLE BIGGER

What did the math teacher say when the bird cage was empty?
> FEBO WED
CaesarKey = 0: FEBO WED
CaesarKey = 10: VURE MUT
CaesarKey = 16: POLY GON
MultiplicativeKey = 9, Additive Key = 4: DARE CAX
MultiplicativeKey = 9, Additive Key = 10: LIZM KIF
MultiplicativeKey = 0, Additive Key = 5: PINA EIB
MultiplicativeKey = 0, Additive Key = 5: HAFS WAT
MultiplicativeKey = 3, Additive Key = 4: JAZM GAR
MultiplicativeKey = 0, Additive Key = 5: NANA AAN
MultiplicativeKey = 23, Additive Key = 12: LUVI OUD
MultiplicativeKey = 23, Additive Key = 16: VEFS YEN
MultiplicativeKey = 23, Additive Key = 11: CLMZ FLU
MultiplicativeKey = 5, Additive Key = 16: DIXK WIN

MultiplicativeKey = 5, Additive Key = 12: JODQ COT
MultiplicativeKey = 5, Additive Key = 4: VAPC OAF
MultiplicativeKey = 17, Additive Key = 0: LOXK MOR
MultiplicativeKey = 17, Additive Key = 24: FIRE GIL

What did Micky Mouse say when he got out of the shower?
> N'R XVZJFPD HQJFS
CaesarKey = 5: I'M SQUEAKY CLEAN

Why did the tomato blush?
> QTRPJHT WT HPL IWT HPAPS SGTHHXCV.
CaesarKey = 15: BECAUSE HE SAW THE SALAD DRESSING.

Should I tell you the peanut butter joke?
> ST, N'R FKWFNI DTZ RNLMY XUWJFI NY.
CaesarKey = 5: NO, I'M AFRAID YOU MIGHT SPREAD IT.

Why should you never take a nap with a pig?
> 06 09 07 05 25 23 09 12 09 01 13 16 16 12 19 11 24 12 09 06 09 08 .
CaesarKey = 5: BECAUSE HE WILL HOG THE BED.

A book never written:
> TQFBLJOH MPVEMZ CZ NJLF SPQIPOF
CaesarKey = 1: SPEAKING LOUDLY BY MIKE ROPHONE
MultiplicativeKey = 9, Additive Key = 15: MDWKOIXC RASTRE NE UIOW JADFAXW
MultiplicativeKey = 9, Additive Key = 6: NEXLPJYD SBTUSF OF VJPX KBEGBYX
MultiplicativeKey = 9, Additive Key = 2: ZQJXBVKP ENFGER AR HVBJ WNQSNKJ
MultiplicativeKey = 21, Additive Key = 25: WHEKIYXO NCGZNA PA SYIE RCHTCXE
MultiplicativeKey = 21, Additive Key = 13: EPMSQGFW VKOHVI XI AGQM ZKPBKFM
MultiplicativeKey = 21, Additive Key = 2: HSPVTJIZ YNRKYL AL DJTP CNSENIP
MultiplicativeKey = 21, Additive Key = 17: KVSYWMLC BQUNBO DO GMWS FQVHQLS
MultiplicativeKey = 21, Additive Key = 16: PAXDBRQH GVZSGT IT LRBX KVAMVQX
MultiplicativeKey = 0, Additive Key = 19: WBCASENQ ZUKVZM HM GESC PUBXUNC
MultiplicativeKey = 0, Additive Key = 19: XCDBTFOR AVLWAN IN HFTD QVCYVOD
MultiplicativeKey = 0, Additive Key = 19: YDECUGPS BWMXBO JO IGUE RWDZWPE
MultiplicativeKey = 3, Additive Key = 20: RQVLXFYN GHJMGT UT PFXV IHQWHYV
MultiplicativeKey = 3, Additive Key = 25: YXCSEMFU NOQTNA BA WMEC POXDOFC
MultiplicativeKey = 3, Additive Key = 16: BAFVHPIX QRTWQD ED ZPHF SRAGRIF
MultiplicativeKey = 3, Additive Key = 12: LKPFRZSH ABDGAN ON JZRP CBKQBSP
MultiplicativeKey = 19, Additive Key = 7: CVEMSWZA DKYTDQ XQ OWSE RKVLKZE
MultiplicativeKey = 19, Additive Key = 2: FYHPVZCD GNBWGT AT RZVH UNYONCH
MultiplicativeKey = 19, Additive Key = 4: JCLTZDGH KRFAKX EX VDZL YRCSRGL
MultiplicativeKey = 19, Additive Key = 25: MFOWCGJK NUIDNA HA YGCO BUFVUJO
MultiplicativeKey = 19, Additive Key = 22: TMVDJNQR UBPKUH OH FNJV IBMCBQV
MultiplicativeKey = 19, Additive Key = 3: UNWEKORS VCQLVI PI GOKW JCNDCRW
MultiplicativeKey = 19, Additive Key = 12: ZSBJPTWX AHVQAN UN LTPB OHSIHWB
MultiplicativeKey = 0, Additive Key = 19: NANNNNAN ANNAAN AN NNNN ANAANAN
MultiplicativeKey = 0, Additive Key = 19: OBOOOOBO BOOBBO BO OOOO BOBBOBO
MultiplicativeKey = 0, Additive Key = 19: RERRRRER ERREER ER RRRR EREERER
MultiplicativeKey = 0, Additive Key = 19: ANAAAANA NAANNA NA AAAA NANNANA
MultiplicativeKey = 0, Additive Key = 19: EREEEERE REERRE RE EEEE RERRERE
MultiplicativeKey = 0, Additive Key = 19: HUHHHHUH UHHUUH UH HHHH UHUUHUH
MultiplicativeKey = 23, Additive Key = 15: QRMWKCJU BAYVBO NO SCKM ZARLAJM
MultiplicativeKey = 23, Additive Key = 18: RSNXLDKV CBZWCP OP TDLN ABSMBKN
MultiplicativeKey = 23, Additive Key = 2: DEZJXPWH ONLIOB AB FPXZ MNEYNWZ
MultiplicativeKey = 23, Additive Key = 11: GHCMASZK RQOLRE DE ISAC PQHBQZC

MultiplicativeKey = 23, Additive Key = 14: HIDNBTAL SRPMSF EF JTBD QRICRAD
MultiplicativeKey = 23, Additive Key = 23: KLGQEWDO VUSPVI HI MWEG TULFUDG
MultiplicativeKey = 11, Additive Key = 1: EZYAIWNK BGQFBO TO UWIY LGZDGNY
MultiplicativeKey = 11, Additive Key = 16: FAZBJXOL CHRGCP UP VXJZ MHAEHOZ
MultiplicativeKey = 11, Additive Key = 25: QLKMUIZW NSCRNA FA GIUK XSLPSZK
MultiplicativeKey = 11, Additive Key = 18: TONPXLCZ QVFUQD ID JLXN AVOSVCN
MultiplicativeKey = 5, Additive Key = 16: LADXZJKT UFBIUH SH PJZD QFAOFKD
MultiplicativeKey = 5, Additive Key = 11: MBEYAKLU VGCJVI TI QKAE RGBPGLE
MultiplicativeKey = 5, Additive Key = 8: XMPJLVWF GRNUGT ET BVLP CRMARWP
MultiplicativeKey = 5, Additive Key = 25: ETWQSCDM NYUBNA LA ICSW JYTHYDW
MultiplicativeKey = 5, Additive Key = 10: HWZTVFGP QBXEQD OD LFVZ MBWKBGZ
MultiplicativeKey = 5, Additive Key = 5: IXAUWGHQ RCYFRE PE MGWA NCXLCHA
MultiplicativeKey = 17, Additive Key = 12: FOVHDJUP ARZYAN EN XJDV IROMRUV
MultiplicativeKey = 17, Additive Key = 24: PYFRNTEZ KBJIKX OX HTNF SBYWBEF

When is a car not a car?
> 25 10 07 16 11 22 22 23 20 16 21 11 16 22 1703 18 03 20 13 11 16 09 14 17 22.
CaesarKey = 3: WHEN IT TURNS INTOA PARKING LOT.
MultiplicativeKey = 21, Additive Key = 11: SVGZ AD DITZY AZDEM JMTKAZQ PED.
MultiplicativeKey = 0, Additive Key = 25: NANA NA ANAAN NAANN ANANNAN ANA.

What do you call a duck who does well in school?
> E AMWI UYEGOIV
CaesarKey = 4: A WISE QUACKER
MultiplicativeKey = 9, Additive Key = 4: A OYCM WIAGEMZ
MultiplicativeKey = 9, Additive Key = 10: I WGKU EQIOMUH
MultiplicativeKey = 21, Additive Key = 4: A GOMU CWAKYUH
MultiplicativeKey = 21, Additive Key = 18: I OWUC KEISGCP
MultiplicativeKey = 0, Additive Key = 4: I GMEK QSIWAKX
MultiplicativeKey = 0, Additive Key = 4: A YEWC IKAOSCP
MultiplicativeKey = 3, Additive Key = 4: A QUGK OYASMKX
MultiplicativeKey = 3, Additive Key = 24: C SWIM QACUOMZ
MultiplicativeKey = 3, Additive Key = 6: I YCOS WGIAUSF
MultiplicativeKey = 19, Additive Key = 4: A IKQS UMAWGSF
MultiplicativeKey = 19, Additive Key = 8: I QSYA CUIEOAN
MultiplicativeKey = 19, Additive Key = 12: Q YAGI KCQMWIV
MultiplicativeKey = 0, Additive Key = 4: A AAAA AAAAAAN
MultiplicativeKey = 0, Additive Key = 4: I IIII IIIIIIV
MultiplicativeKey = 23, Additive Key = 4: A KGUQ MCAIOQD
MultiplicativeKey = 23, Additive Key = 2: I SOCY UKIQWYL
MultiplicativeKey = 11, Additive Key = 4: A CWEY SQAMIYL
MultiplicativeKey = 11, Additive Key = 20: I KEMG AYIUQGT
MultiplicativeKey = 5, Additive Key = 16: I CUWO GMIYKOB
MultiplicativeKey = 5, Additive Key = 4: A UMOG YEAQCGT
MultiplicativeKey = 17, Additive Key = 4: A MCYO ESAUWOB
MultiplicativeKey = 17, Additive Key = 24: I UKGW MAICEWJ

What question can you not say yes to?
> RIV PFL JCVVGZEX?
CaesarKey = 17: ARE YOU SLEEPING?
MultiplicativeKey = 5, Additive Key = 5: SLY CAW GPYYVEFO?

What's the richest type of air?
> W XEHHEKJWENA!

CaesarKey = 14: I JQTTQWVIQZM!
CaesarKey = 22: A BILLIONAIRE!
MultiplicativeKey = 9, Additive Key = 22: A DYHHYQNAYZM!
MultiplicativeKey = 9, Additive Key = 2: I LGPPGYVIGHU!
MultiplicativeKey = 21, Additive Key = 10: I NWLLWAVIWPC!
MultiplicativeKey = 21, Additive Key = 22: A FODDOSNAOHU!
MultiplicativeKey = 0, Additive Key = 22: A HEZZEUNAEPC!
MultiplicativeKey = 0, Additive Key = 22: I PMHHMCVIMXK!
MultiplicativeKey = 3, Additive Key = 22: A JUVVUWNAUXK!
MultiplicativeKey = 3, Additive Key = 24: I RCDDCEVICFS!
MultiplicativeKey = 19, Additive Key = 0: I TSZZSGVISNA!
MultiplicativeKey = 19, Additive Key = 22: A LKRRKYNAKFS!
MultiplicativeKey = 0, Additive Key = 22: A NANNAANAANA!
MultiplicativeKey = 0, Additive Key = 22: I VIVVIIVIIVI!
MultiplicativeKey = 23, Additive Key = 22: A RGFFGENAGDQ!
MultiplicativeKey = 23, Additive Key = 20: I ZONNOMVIOLY!
MultiplicativeKey = 11, Additive Key = 12: I BEJJEOVIETG!
MultiplicativeKey = 11, Additive Key = 22: A TWBBWGNAWLY!
MultiplicativeKey = 5, Additive Key = 22: A VMXXMINAMTG!
MultiplicativeKey = 5, Additive Key = 8: I DUFFUQVIUBO!
MultiplicativeKey = 17, Additive Key = 22: A XCTTCKNACBO!
MultiplicativeKey = 17, Additive Key = 16: I FKBBKSVIKJW!

The forest is two miles across. How far can you go into it?
> QPG OKNG. CHVGT VJCV AQW YKNN DG IQKPI QWV.
CaesarKey = 2: ONE MILE. AFTER THAT YOU WILL BE GOING OUT.

Is there a word in the English language that contains all the vowels?
> XQTXHVWLRQDEOB
CaesarKey = 3: UNQUESTIONABLY

What has eighteen legs and catches flies?
> 12 13 12 04 16 13 12 23 23 05 16 12 24
CaesarKey = 12: A BASEBALL TEAM
When is an elevator not an elevator?
> GROX SD SC QYSXQ NYGX
CaesarKey = 10: WHEN IT IS GOING DOWN

Why is a poor joke like an unsharpened pencil?
> QTRPJHT XI WPH CD EDXCI
CaesarKey = 15: BECAUSE IT HAS NO POINT

What do you call a pig with three eyes?
> U JCCCA!
CaesarKey = 12: I XQQQO!
CaesarKey = 20: A PIIIG!
MultiplicativeKey = 9, Additive Key = 0: I BGGGA!
MultiplicativeKey = 9, Additive Key = 20: A TYYYS!
MultiplicativeKey = 21, Additive Key = 20: A XOOOE!
MultiplicativeKey = 21, Additive Key = 8: I FWWWM!
MultiplicativeKey = 0, Additive Key = 20: A BEEEQ!
MultiplicativeKey = 0, Additive Key = 20: I JMMMY!
MultiplicativeKey = 3, Additive Key = 20: A FUUUC!
MultiplicativeKey = 3, Additive Key = 22: I NCCCK!
MultiplicativeKey = 19, Additive Key = 20: A JKKKO!
MultiplicativeKey = 19, Additive Key = 24: I RSSSW!

MultiplicativeKey = 0, Additive Key = 20: A NAAAA!
MultiplicativeKey = 0, Additive Key = 20: I VIIII!
MultiplicativeKey = 23, Additive Key = 18: I DOOOG!
MultiplicativeKey = 23, Additive Key = 20: A VGGGY!
MultiplicativeKey = 11, Additive Key = 20: A ZWWWK!
MultiplicativeKey = 11, Additive Key = 10: I HEEES!
MultiplicativeKey = 5, Additive Key = 6: I LUUUE!
MultiplicativeKey = 5, Additive Key = 20: A DMMMW!
MultiplicativeKey = 17, Additive Key = 20: A HCCCI!
MultiplicativeKey = 17, Additive Key = 14: I PKKKQ!

I can be cracked, I can be made.I can be told, I can be played.What am I?
> A BQEM
CaesarKey = 0: A BQEM
CaesarKey = 18: I JYMU
MultiplicativeKey = 3, Additive Key = 0: A JOKE

Why are cooks cruel?
> FUKAWMU RJUQ FUAR RJU UEEM ANP GJOX RJU KHUAI.
CaesarKey = 9: WLBRNDL IALH WLRI IAL LVVD REG XAFO IAL BYLRZ.
MultiplicativeKey = 5, Additive Key = 0: BECAUSE THEY BEAT THE EGGS AND WHIP THE CREAM.

What is it that is always coming but never arrives?
> BYCYFFYI. IZSN KB AFFKXSQ, KB KQ BYHAE.
CaesarKey = 10: ROSOVVOY. YPID AR QVVANIG, AR AG ROXQU.
MultiplicativeKey = 11, Additive Key = 0: TOMORROW. WHEN IT ARRIVES, IT IS TODAY.

Why is the figure 9 like a peacock?
> TUCKIQU AEZVGIZ K ZKEF EZ EQ XGZVEXM.
CaesarKey = 10: JKSAYGK QUPLWYP A PAUV UP UG NWPLUNC.
MultiplicativeKey = 9, Additive Key = 10: BECAUSE WITHOUT A TAIL IT IS NOTHING.

What colors would you paint the sun and the wind?
> PTU YGR HIYU ERD PTU OKRD VJUO.
CaesarKey = 5: KOP TBM CDTP ZMY KOP JFMY QEPJ.
CaesarKey = 19: WAB FNY OPFB LYK WAB VRYK CQBV.
MultiplicativeKey = 17, Additive Key = 4: THE SUN ROSE AND THE WIND BLEW.

Why did the chicken cross the playground?
> QL DBQ QL QEB LQEBO PIFAB!
CaesarKey = 23: TO GET TO THE OTHER SLIDE!

What is the most slippery country in the world?
> XIVVTV!
CaesarKey = 17: GREECE!

What's the difference between a cat and a comma?
> O QOH VOG QZOKG OH HVS SBR CT DOKG; O QCAAO WG O DOIGS OH HVS SBR CT O QZOIGS.
CaesarKey = 14: A CAT HAS CLAWS AT THE END OF PAWS; A COMMA IS A PAUSE AT THE END OF A CLAUSE.

Why did the forgetful chicken cross the road?
> HC USH HC HVS CHVSF GWRS -- SF, BC -- HC UC GVCDDWBU -- BC, BCH HVOH SWHVSF -- ROBU WH!
CaesarKey = 14: TO GET TO THE OTHER SIDE -- ER, NO -- TO GO SHOPPING -- NO, NOT THAT EITHER -- DANG IT!

Who is she?
> X CBJXIB !
CaesarKey = 23: A FEMALE !

How does NASA organize its company parties?
> WKHB SODQHW.
CaesarKey = 3: THEY PLANET.

Give me food, and I will live; give me water, and I will die. What am I?
> 07 12 15 24 11!
CaesarKey = 7: A FIRE!
MultiplicativeKey = 0, Additive Key = 7: A NANA!
MultiplicativeKey = 23, Additive Key = 5: I POLY!

Did you hear the joke about peanut butter?
> V'Z ABG GRYYVAT, LBH ZVTUG FCERNQ VG!
CaesarKey = 13: I'M NOT TELLING, YOU MIGHT SPREAD IT!

Why should you put cheese on your computer?
> WR IHHG WKH PRXVH.
CaesarKey = 3: TO FEED THE MOUSE.

How many people does it take to screw in a lightbulb?
> ZCU. UTK ZU HAE G RGJJKX, UTK ZU ZAXT ZNK ROMNZHARH!
CaesarKey = 6: TWO. ONE TO BUY A LADDER, ONE TO TURN THE LIGHTBULB!

When are 2 heads not better than 1?
> CNKT ZNKE JUT'Z IUUVKXGZK.
CaesarKey = 6: WHEN THEY DON'T COOPERATE.