M. Caner TOL
859565356

09/08/2019

# ECE/CS 578 Assignment 1

* Due: 11:59 pm on Sept 14, 2019 (submit a soft copy via Canvas)

1. The ciphertext printed below was encrypted using a substitution cipher. The objective is to decrypt the ciphertext without knowledge of the key.
   a. Provide the relative frequency of all letters A...Z in the ciphertext.

      {**'A'**: 0.13928, **'B'**: 0.09285, **'C'**: 0.07985, **'D'**: 0.07057, **'E'**: 0.05385, **'F'**: 0.07707,
      **'G'**: 0.065, **'H'**: 0.04178, **'I'**: 0.06964, **'J'**: 0.03714, **'K'**: 0.04364, **'L'**: 0.04643, **'M'**: 0.03435,
      **'N'**: 0.02228, **'O'**: 0.01764, **'P'**: 0.01764, **'Q'**: 0.02136, **'R'**: 0.01393, **'S'**: 0.02228,
      **'T'**: 0.00836, **'U'**: 0.01393, **'V'**: 0.00836, **'W'**: 0.0, **'X'**: 0.0, **'Y'**: 0.00279, **'Z'**: 0.0}

   b. Decrypt the ciphertext with help of the relative letter frequency of the English language (e.g., search Wikipedia for letter frequency analysis). Note that the text is relatively short and might not completely fulfill the given frequencies from the table.

      'ELECTRICAL AND COMPUTER ENGINEERS DEVELOP AND CREATE PRODUCTS THAT CHANGE THE WORLD AND MAKE OUR LIVES EASIER THE CELL PHONES WE DEPEND ON THE COMPUTERS USED IN NATIONAL SECURITY AND THE ELECTRICAL SYSTEMS THAT MAKE OUR CARS OPERATE WERE ALL CREATED BY ELECTRICAL AND COMPUTER ENGINEERS AT WPI WE KEEP THAT PROGRESS MOVING FORWARD WITH OUR INNOVATIVE RESEARCH AND OUT OF THE BOX APPROACHES THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING AT WPI CHALLENGES STUDENTS TO PUSH THEMSELVES TO UNDERSTAND SOCIETYS AND TECHNOLOGYS COMPLEX ISSUES IN A BROADER CONTEXT THAN WHATS IN FRONT OF THEM WE WANT OUR STUDENTS WHETHER THEY ARE EARNING AN UNDERGRADUATE MINOR OR A DOCTORATE TO TACKLE SOCIETYS MOST PRESSING PROBLEMS AND UNCOVER NEW WAYS OF SOLVING THEM WHETHER ITS DEVELOPING SYSTEMS THAT CAN LOCATE FIREFIGHTERS IN THE MIDDLE OF A BURNING BUILDING OR CREATING NEUROPROSTHETICS THAT LOOK AND FUNCTION LIKE NATURAL LIMBS OUR FACULTY AND STUDENTS ARE AT THE FRONT EDGE OF REMARKABLE INNOVATION WHILE ADVANCING TECHNOLOGIES IS AT OUR CORE WE ALSO TAKE HUMAN CONNECTIONS VERY SERIOUSLY IN ECE WE PRIDE OURSELVES ON THE FAMILY LIKE ATMOSPHERE WE CULTIVATE FACULTY STUDENTS AND STAFF ENCOURAGE EACH OTHERS EVERY SUCCESS AND ARE THERE FOR THE CHALLENGES BOTH IN THE CLASSROOM AND IN LIFE'

   c. Find the Plaintext/Ciphertext letter pairs, alphabetized by plaintext.
      {**'A'**: **'E'**, **'B'**: **'T'**, **'C'**: **'A'**, **'D'**: **'O'**, **'E'**: **'I'**, **'F'**: **'N'**, **'G'**: **'S'**, **'H'**: **'H'**, **'I'**: **'R'**, **'J'**: **'D'**, **'K'**: **'L'**, **'L'**: **'C'**,
      **'M'**: **'U'**, **'N'**: **'M'**, **'O'**: **'W'**, **'P'**: **'F'**, **'Q'**: **'G'**, **'R'**: **'Y'**, **'S'**: **'P'**, **'T'**: **'B'**, **'U'**: **'V'**, **'V'**: **'K'**, **'W'**: **'Z'**,
      **'X'**: **'Z'**, **'Y'**: **'X'**, **'Z'**: **'Z'**}

d. Provide letter frequency for the given plaintext.

{**'A'**: 0.07985, **'B'**: 0.00836, **'C'**: 0.04643, **'D'**: 0.03714, **'E'**: 0.13928, **'F'**: 0.01764, **'G'**: 0.02136, **'H'**: 0.04178, **'I'**: 0.05385, **'J'**: 0.0, **'K'**: 0.00836, **'L'**: 0.04364, **'M'**: 0.02228, **'N'**: 0.07707, **'O'**: 0.07057, **'P'**: 0.02228, **'Q'**: 0.0, **'R'**: 0.06964, **'S'**: 0.065, **'T'**: 0.09285, **'U'**: 0.03435, **'V'**: 0.01393, **'W'**: 0.01764, **'X'**: 0.00279, **'Y'**: 0.01393, **'Z'**: 0.0}

Ciphertext:

AKALBIELCK CFJ LDNSMBAI AFQEFAAIG JAUAKDS CFJ LIACBA SIDJMLBG BHCB LHCFQA BHA ODIKJ CFJ NCVA DMI KEUAG ACGEAI BHA LAKK SHDFAG OA JASAFJ DF BHA LDNSMBAIG MGAJ EF FCBEDFCK GALMIEBR CFJ BHA AKALBIELCK GRGBANG BHCB NCVA DMI LCIG DSAICBA OAIA CKK LIACBAJ TR AKALBIELCK CFJ LDNSMBAI AFQEFAAIG CB OSE OA VAAS BHCB SIDQIAGG NDUEFQ PDIOCIJ OEBH DMI EFFDUCBEUA IAGACILH CFJ DMB-DP-BHA TDY CSSIDCLHAG BHA JASCIBNAFB DP AKALBIELCK CFJ LDNSMBAI AFQEFAAIEFQ CB OSE LHCKKAFQAG GBMJAFBG BD SMGH BHANGAKUAG BD MFJAIGBCFJ GDLEABRG CFJ BALHFDKDQRG LDNSKAY EGGMAG EF C TIDCJAI LDFBAYB BHCF OHCBG EF PIDFB DP BHAN OA OCFB DMI GBMJAFBG OHABHAI BHAR CIA ACIFEFQ CF MFJAIQICJMCBA NEFDI DI C JDLBDICBA BD BCLVKA GDLEABRG NDGB SIAGGEFQ SIDTKANG CFJ MFLDUAI FAO OCRG DP GDKUEFQ BHAN OHABHAI EBG JAUAKDSEFQ GRGBANG BHCB LCF KDLCBA PEIAPEQHBAIG EF BHA NEJJKA DP C TMIFEFQ TMEKJEFQ DI LIACBEFQ FAMIDSIDGBHABELG BHCB KDDV CFJ PMFLBEDF KEVA FCBMICK KENTG DMI PCLMKBR CFJ GBMJAFBG CIA CB BHA PIDFB AJQA DP IANCIVCTKA EFFDUCBEDF OHEKA CJUCFLEFQ BALHFDKDQEAG EG CB DMI LDIA OA CKGD BCVA HMNCF LDFFALBEDFG UAIR GAIEDMGKR EF ALA OA SIEJA DMIGAKUAG DF BHA PCNEKR-KEVA CBNDGSHAIA OA LMKBEUCBA; PCLMKBR GBMJAFBG CFJ GBCPP AFLDMICQA ACLH DBHAIG AUAIR GMLLAGG CFJ CIA BHAIA PDI BHA LHCKKAFQAG TDBH EF BHA LKCGGIDDN CFJ EF KEPA

```
cipher_text = 'AKALBIELCK CFJ LDNSMBAI AFQEFAAIG JAUAKDS CFJ LIACBA ' \
              'SIDJMLBG BHCB LHCFQA BHA ODIKJ CFJ NCVA DMI KEUAG ACGEAI ' \
              'BHA LAKK SHDFAG OA JASAFJ DF BHA LDNSMBAIG MGAJ EF FCBEDFCK ' \
              'GALMIEBR CFJ BHA AKALBIELCK GRGBANG BHCB NCVA DMI LCIG ' \
              'DSAICBA OAIA CKK LIACBAJ TR AKALBIELCK CFJ LDNSMBAI AFQEFAAIG ' \
              'CB OSE OA VAAS BHCB SIDQIAGG NDUEFQ PDIOCIJ OEBH DMI EFFDUCBEUA ' \
              'IAGACILH CFJ DMB DP BHA TDY CSSIDCLHAG BHA JASCIBNAFB DP AKALBIELCK ' \
              'CFJ LDNSMBAI AFQEFAAIEFQ CB OSE LHCKKAFQAG GBMJAFBG BD SMGH ' \
              'BHANGAKUAG BD MFJAIGBCFJ GDLEABRG CFJ BALHFDKDQRG LDNSKAY EGGMAG ' \
              'EF C TIDCJAI LDFBAYB BHCF OHCBG EF PIDFB DP BHAN OA OCFB DMI ' \
              'GBMJAFBG OHABHAI BHAR CIA ACIFEFQ CF MFJAIQICJMCBA NEFDI DI C ' \
              'JDLBDICBA BD BCLVKA GDLEABRG NDGB SIAGGEFQ SIDTKANG CFJ MFLDUAI ' \
              'FAO OCRG DP GDKUEFQ BHAN OHABHAI EBG JAUAKDSEFQ GRGBANG BHCB LCF ' \
              'KDLCBA PEIAPEQHBAIG EF BHA NEJJKA DP C TMIFEFQ TMEKJEFQ DI LIACBEFQ ' \
              'FAMIDSIDGBHABELG BHCB KDDV CFJ PMFLBEDF KEVA FCBMICK KENTG DMI PCLMKBR ' \
              'CFJ GBMJAFBG CIA CB BHA PIDFB AJQA DP IANCIVCTKA EFFDUCBEDF OHEKA ' \
              'CJUCFLEFQ BALHFDKDQEAG EG CB DMI LDIA OA CKGD BCVA HMNCF LDFFALBEDFG ' \
              'UAIR GAIEDMGKR EF ALA OA SIEJA DMIGAKUAG DF BHA PCNEKR KEVA CBNDGSHAIA ' \
              'OA LMKBEUCBA PCLMKBR GBMJAFBG CFJ GBCPP AFLDMICQA ACLH DBHAIG AUAIR ' \
              'GMLLAGG CFJ CIA BHAIA PDI BHA LHCKKAFQAG TDBH EF BHA LKCGGIDDN CFJ EF KEPA'

alphabet = {'A': 0, 'B': 0, 'C': 0, 'D': 0, 'E': 0, 'F': 0, 'G': 0, 'H': 0, 'I': 0,
            'J': 0, 'K': 0, 'L': 0, 'M': 0, 'N': 0, 'O': 0, 'P': 0, 'Q': 0, 'R': 0,
            'S': 0, 'T': 0, 'U': 0, 'V': 0, 'W': 0, 'X': 0, 'Y': 0, 'Z': 0}
words = []
[words.append(list(cipher_text.split()[idx])) for idx in range(len(cipher_text.split()))]
count=0 # total number of letters in the text
for word in words:
```

```
29      for letter in word:
30          alphabet[letter] += 1
31          count += 1
32  for letter in alphabet:
33      alphabet[letter] = round(alphabet[letter]/count, 5)
34  # alphabet is now a dictionary of (letter : relative frequency value)
    pairs.
35  print(alphabet)
36
37  # a.    Provide the relative frequency of all letters A...Z in the
    ciphertext.********
38  # Output: {'A': 0.13928, 'B': 0.09285, 'C': 0.07985, 'D': 0.07057, 'E': 0
    .05385, 'F': 0.07707,
39  # 'G': 0.065, 'H': 0.04178, 'I': 0.06964, 'J': 0.03714, 'K': 0.04364, 'L
    ': 0.04643, 'M': 0.03435,
40  # 'N': 0.02228, 'O': 0.01764, 'P': 0.01764, 'Q': 0.02136, 'R': 0.01393, '
    S': 0.02228,
41  # 'T': 0.00836, 'U': 0.01393, 'V': 0.00836, 'W': 0.0, 'X': 0.0, 'Y': 0.
    00279, 'Z': 0.0}
42
43
44  # English letter frequency [Wikipedia https://en.wikipedia.org/wiki/
    Letter_frequency ]:
45  D = {'A': 0.08167, 'B': 0.01492, 'C': 0.02782, 'D': 0.04253, 'E': 0.12702
    , 'F': 0.02228, 'G': 0.02015, 'H': 0.06094,
46       'I': 0.06966, 'J': 0.00153, 'K': 0.00772, 'L': 0.04025, 'M': 0.02406
    , 'N': 0.06749, 'O': 0.07507, 'P': 0.01929,
47       'Q': 0.00095, 'R': 0.05987, 'S': 0.06327, 'T': 0.09056, 'U': 0.02758
    , 'V': 0.00978, 'W': 0.02360, 'X': 0.00150,
48       'Y': 0.01974, 'Z': 0.00074}
49  plain=1
50  for letter in alphabet:
51      for key in D:
52          if abs(alphabet[letter]-D[key])< plain:
53              target = key
54              plain = abs(alphabet[letter]-D[key])
55      alphabet[letter] = target
56      plain = 1
57  print(alphabet)
58  # Output:
59  #{'A': 'E', 'B': 'T', 'C': 'A', 'D': 'I', 'E': 'R', 'F': 'O', 'G': 'S', '
    H': 'D', 'I': 'I',
60  # 'J': 'L', 'K': 'D', 'L': 'D', 'M': 'L', 'N': 'F', 'O': 'P', 'P': 'P', '
    Q': 'F', 'R': 'B',
61  # 'S': 'F', 'T': 'K', 'U': 'B', 'V': 'K', 'W': 'Z', 'X': 'Z', 'Y': 'J', '
    Z': 'Z'}
62
63  #c. Find the Plaintext/Ciphertext letter pairs, alphabetized by plaintext
    .****
64  # Modify substitution until a reasonable plaintext
```

```
alphabet = {'A': 'E', 'B': 'T', 'C': 'A', 'D': 'O', 'E': 'I', 'F': 'N', 'G': 'S', 'H': 'H', 'I': 'R', 'J': 'D',
            'K': 'L', 'L': 'C', 'M': 'U', 'N': 'M', 'O': 'W', 'P': 'F', 'Q': 'G', 'R': 'Y', 'S': 'P', 'T': 'B',
            'U': 'V', 'V': 'K', 'W': 'Z', 'X': 'Z', 'Y': 'X', 'Z': 'Z'}


for word in words:
    for i,letter in enumerate(word):
        word[i]=alphabet[letter]
temp = []
plainText = []
[temp.append(''.join(word)) for word in words]
[plainText.append(' '.join(temp))]
print(plainText)
# Output before modifying the substitution:
# ['EDEDTIRDAD AOL DIFFLTEI EOFROEEIS LEBEDIF AOL DIEATE FIILLDTS TDAT DDAOFE TDE PIIDL AOL
# FAKE ILI DRBES EASREI TDE DEDD FDIOES PE LEFEOL IO TDE DIFFLTEIS LSEL RO OATRIOAD SEDLIRTB
# AOL TDE EDEDTIRDAD SBSTEFS TDAT FAKE ILI DAIS IFEIATE PEIE ADD DIEATEL KB EDEDTIRDAD AOL
# DIFFLTEI EOFROEEIS AT PFR PE KEEF TDAT FIIFIESS FIBROF PIIPAIL PRTD ILI ROOIBATRBE IESEAIDD
# AOL ILT IP TDE KIJ AFFIIADDES TDE LEFAITFEOT IP EDEDTIRDAD AOL DIFFLTEI EOFROEEIROF AT PFR
# DDADDEOFES STLLEOTS TI FLSD TDEFSEDBES TI LOLEISTAOL SIDRETBS AOL TEDDOIDIFBS DIFFDEJ RSSLES
# RO A KIIALEI DIOTEJT TDAO PDATS RO PIIOT IP TDEF PE PAOT ILI STLLEOTS PDETDEI TDEB AIE EAIOROF
# AO LOLEIFIALLATE FROII II A LIDTIIATE TI TADKDE SIDRETBS FIST FIESSROF FIIKDEFS AOL LODIBEI
# OEP PABS IP SIDBROF TDEF PDETDEI RTS LEBEDIFROF SBSTEFS TDAT DAO DIDATE PRIEPRFDTEIS RO TDE
# FRLLDE IP A KLIOROF KLRDLROF II DIEATROF OELIIFIISTDETRDS TDAT DIIK AOL PLODTRIO DRKE OATLIAD
# DRFKS ILI PADLDTB AOL STLLEOTS AIE AT TDE PIIOT ELFE IP IEFAIKAKDE ROOIBATRIO PDRDE ALBAODROF
# TEDDOIDIFRES RS AT ILI DIIE PE ADSI TAKE DLFAO DIOOEDTRIOS BEIB SEIRILSDB RO EDE PE FIRLE ILISEDBES
# IO TDE PAFRDB DRKE ATFISFDEIE PE DLDTRBATE PADLDTB STLLEOTS AOL STAPP EODILIAFE EADD ITDEIS EBEIB
# SLDDESS AOL AIE TDEIE PII TDE DDADDEOFES KITD RO TDE DDASSIIIF AOL RO DRPE'

# b.    Decrypt the ciphertext with help of the relative letter frequency of the
# English language (e.g., search Wikipedia for letter frequency analysis ).
# Note that the text is relatively short and might not completely
```

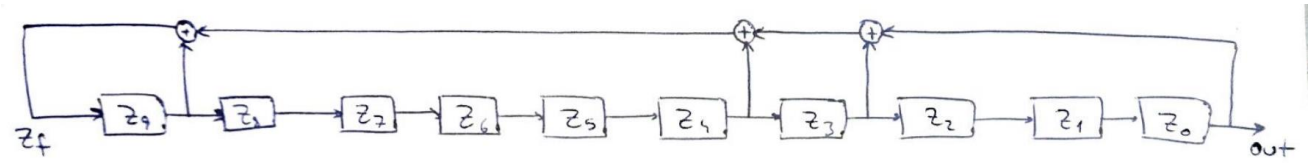
```
96 fulfill the
97 # given frequencies from the table.
98
99 # After modifying the alphabet
100 # 'ELECTRICAL AND COMPUTER ENGINEERS DEVELOP AND CREATE PRODUCTS THAT
    CHANGE THE WORLD AND MAKE
101 # OUR LIVES EASIER THE CELL PHONES WE DEPEND ON THE COMPUTERS USED IN
    NATIONAL SECURITY AND THE
102 # ELECTRICAL SYSTEMS THAT MAKE OUR CARS OPERATE WERE ALL CREATED BY
    ELECTRICAL AND COMPUTER ENGINEERS
103 # AT WPI WE KEEP THAT PROGRESS MOVING FORWARD WITH OUR INNOVATIVE
    RESEARCH AND OUT OF THE BOX APPROACHES
104 # THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING AT WPI
    CHALLENGES STUDENTS TO PUSH THEMSELVES
105 # TO UNDERSTAND SOCIETYS AND TECHNOLOGYS COMPLEX ISSUES IN A BROADER
    CONTEXT THAN WHATS IN FRONT OF THEM
106 # WE WANT OUR STUDENTS WHETHER THEY ARE EARNING AN UNDERGRADUATE MINOR
    OR A DOCTORATE TO TACKLE SOCIETYS
107 # MOST PRESSING PROBLEMS AND UNCOVER NEW WAYS OF SOLVING THEM WHETHER
    ITS DEVELOPING SYSTEMS THAT CAN LOCATE
108 # FIREFIGHTERS IN THE MIDDLE OF A BURNING BUILDING OR CREATING
    NEUROPROSTHETICS THAT LOOK AND FUNCTION LIKE
109 # NATURAL LIMBS OUR FACULTY AND STUDENTS ARE AT THE FRONT EDGE OF
    REMARKABLE INNOVATION WHILE ADVANCING TECHNOLOGIES
110 # IS AT OUR CORE WE ALSO TAKE HUMAN CONNECTIONS VERY SERIOUSLY IN ECE WE
     PRIDE OURSELVES ON THE FAMILY LIKE
111 # ATMOSPHERE WE CULTIVATE FACULTY STUDENTS AND STAFF ENCOURAGE EACH
    OTHERS EVERY SUCCESS AND ARE THERE FOR THE
112 # CHALLENGES BOTH IN THE CLASSROOM AND IN LIFE'
113
114
115 # Letter frequency analysis for the plaintext
116 p_alphabet = {'A': 0, 'B': 0, 'C': 0, 'D': 0, 'E': 0, 'F': 0, 'G': 0, 'H
    ': 0, 'I': 0,
117             'J': 0, 'K': 0, 'L': 0, 'M': 0, 'N': 0, 'O': 0, 'P': 0, 'Q':
     0, 'R': 0,
118             'S': 0, 'T': 0, 'U': 0, 'V': 0, 'W': 0, 'X': 0, 'Y': 0, 'Z':
     0}
119 count=0 # total number of letters in the text
120 for word in words:
121     for letter in word:
122         p_alphabet[letter] += 1
123         count += 1
124 for letter in p_alphabet:
125     p_alphabet[letter] = round(p_alphabet[letter]/count, 5)
126
127 print(p_alphabet)
128
129 #d. Provide letter frequency for the given plaintext.********
130 # Output:
```

```
131 # {'A': 0.07985, 'B': 0.00836, 'C': 0.04643, 'D': 0.03714, 'E': 0.13928
    , 'F': 0.01764,
132 # 'G': 0.02136, 'H': 0.04178, 'I': 0.05385, 'J': 0.0, 'K': 0.00836, 'L
    ': 0.04364,
133 # 'M': 0.02228, 'N': 0.07707, 'O': 0.07057, 'P': 0.02228, 'Q': 0.0, 'R
    ': 0.06964,
134 # 'S': 0.065, 'T': 0.09285, 'U': 0.03435, 'V': 0.01393, 'W': 0.01764, 'X
    ': 0.00279,
135 # 'Y': 0.01393, 'Z': 0.0}
136
137
```

M. Caner TOL
859565356

09/08/2019

2. An LFSR is given by $(m, (C_0, C_1, \cdots, C_9), (Z_0, Z_1, \cdots, Z_9) = (10, (1,0,0,1,1,0,0,0,0,1), (0,0,0,1,1,0,1,0,0,0))$.
   a. Draw a circuit diagram for the given LFSR.




   b. Compute first 512 bits of the output bit stream

```
Z = [0, 0, 0, 1, 1, 0, 1, 0, 0, 0]
out = []
for k in range(512):
    Zf = Z[0] ^ Z[3] ^ Z[4] ^ Z[9]
    out.append(Z.pop(0))
    Z.append(Zf)
print(out)
```

```
0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1,
0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1,
0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,
0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0,
1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,
1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1,
0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1
```

M. Caner TOL
859565356

09/08/2019

c. Apply the following test to output bits

```
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <bitstream.txt>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
  1   0   0   0   0   0   0   0   0   0     ----       0/1       Frequency
  0   0   0   0   0   0   1   0   0   0     ----       1/1       BlockFrequency
  1   0   0   0   0   0   0   0   0   0     ----       0/1       Runs


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 0 for a
sample size = 1 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
              FILE = bitstream.txt           ALPHA = 0.0100

              BITSREAD = 512 0s = 314 1s = 198
```

i. Apply the NIST Frequency Test* to the given sequence.

```
                              FREQUENCY TEST
                    ----------------------------------------------
                    COMPUTATIONAL INFORMATION:
                    ----------------------------------------------
                    (a) The nth partial sum = -116
                    (b) S_n/n               = -0.226562
                    ----------------------------------------------
FAILURE             p_value = 0.000000
```

ii. Apply the NIST Frequency Test within a Block* to the given sequence. Use the block size 16.

```
                BLOCK FREQUENCY TEST
               ----------------------------------------------
               COMPUTATIONAL INFORMATION:
               ----------------------------------------------
               (a) Chi^2           = 29.000000
               (b) # of substrings = 32
               (c) block length    = 16
               (d) Note: 0 bits were discarded.
               ----------------------------------------------
SUCCESS        p_value = 0.619163
```

iii. Apply the NIST Runs Test* to the given sequence.

```
                    RUNS TEST
          ------------------------------------------
          PI ESTIMATOR CRITERIA NOT MET! PI = 0.386719
```

d. Encrypt the following plaintext using the bit stream generated above.
P=`11101100000110111011010011111010000`
K='00011010000100110100101001001010001'
C='11110110000010001111111010110000001

* You can find NIST Randomness Tests at
http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html