Problem 3

We have three parts:

- Imports, global function definitions, and text import.
  - Import required libraries (*i.e. time, csv, etc*)
  - Define any required global functions
  - Import the text to work with from its .txt file
- Work for the LZ-Compression part
- Work for the LZ-DEcompression part

Thus, we now move to imports, global function definitions, and text import.

Imports, Global Function Definitions, and Text-File Import.

We start with library imports.

Library Imports

We require the **time**

```
In [1]:  import time
```

the **csv**

```
In [2]:  import csv
```

and the **requests** libraries

```
In [3]:  import requests
```

Then we move on to global function definitions.

Global Function Definitions

We require four global functions. These are

- A text-file reader
- A text-file writer
- A csv-file reader
- A csv-file writer

Text-File Reader

We start with a text-file reader

```
In [4]: def fileRDR(filename):
            with open(filename, 'r') as myTextFileIn:
                myTextIn = myTextFileIn.read();

                myTextFileIn.close()


            return myTextIn
```

Text-File Writer

We continue with a text-file writer

```
In [5]: def fileWTR(filename, strToWrite):
            with open(filename, 'w') as myTextFileOut:
                myTextFileOut.write(strToWrite)

                myTextFileOut.close()


            return None
```

CSV-File reader

We follow that with a csv-file reader

```
In [6]: def csvFileRDR(filename):
            csvOUT = []

            with open(filename, 'r') as myCSVfileIn:
                csvReader = csv.reader(x for x in myCSVfileIn)

                for row in csvReader:
                    temp = row
                    csvOUT.append(temp)

                myCSVfileIn.close()


            return csvOUT
```

CSV-File Writer

We finish with a csv-file writer

```
In [7]: def csvFileWTR(filename, arrayToWrite):
            with open(filename, 'w', newline='') as myCSVfileOut:
                csvWriter = csv.writer(myCSVfileOut, delimiter=',',
                                   quotechar=' ', quoting=csv.QUOTE_MINIMAL)
                                       #dialect='excel')
                                       #delimiter=',', quotechar='|', quoting=csv.QUOTE_MI
        NIMAL)

                for row in arrayToWrite:
                    csvWriter.writerow(row)

                myCSVfileOut.close()

            return None
```

Text Import

We will now import two different text-files for use in this problem.

Tom Sawyer

The first, is "*Tom Sawyer*"

```
In [8]: textInA = fileRDR('../Text-Files/sawyer-ascii.txt')
```

King James Bible

The second in the "*King James version of the Bible*"

```
In [9]: textInB = fileRDR('../Text-Files/kingJames-ascii.txt')
```

Having finished importing libraries, defining global functions, and importing the files we are going to work with, we move on to the work for LZ-Compression

LZ-Compression

We will first define a method for compression a text file using the Lempel-Ziv Compression routine specified in the book.

In [10]:

```python
def lzCompr(textIn):

    myDict = dict()
    myAns = []
    myResult = ''
    myComp = ''


    posCTR = 1
    word = ''
    for char in textIn:
        wordNchar = word + char

        if (wordNchar in myDict):
            word = wordNchar
        else:
            myDict[wordNchar] = posCTR
            posCTR += 1

            if (len(wordNchar) == 1):
                myAns.append([0, char])
            else:
                anINT = myDict[word]
                myAns.append([anINT, char])

            word = ''


    if (word):
        anINT = myDict[word]
        myAns.append([anINT])


    for row in myAns:
        for col in row:
            myResult = myResult + str(col)


    for row in myAns:
        tst = 0

        for col in row:

            if tst == 0:
                myComp = myComp + str(col) + ','
            else:
                myComp = myComp + str(col) + ';'

            tst += 1


    myOut = [myAns, myResult, myComp]

    return myOut
```

Alternate Function

We also define this alternate, and slightly simpler, function

```
In [11]:  def lzComprA(textIn):

              myDict = dict()
              myAns = []


              posCTR = 1
              word = ''
              for char in textIn:
                  wordNchar = word + char

                  if (wordNchar in myDict):
                      word = wordNchar
                  else:
                      myDict[wordNchar] = posCTR
                      posCTR += 1

                      if (len(wordNchar) == 1):
                          myAns.append([0, char])
                      else:
                          anINT = myDict[word]
                          myAns.append([anINT, char])

                      word = ''


              if (word):
                  anINT = myDict[word]
                  myAns.append([anINT])


              return myAns
```

Tom Sawyer

Now, we can run and time the compression routine for *Tom Sawyer*.

Initial Verion

By using the first routine we defined

```
In [12]:  t0 = time.time()
          compTextArrA = lzCompr(textInA)
          t1 = time.time()
```

To get the results

```
In [13]:  formCompText = compTextArrA[2]
          noFormCompText = compTextArrA[1]

          tTot = t1 - t0
          origTextLEN = len(textInA)
          formCompTextLEN = len(formCompText)
          noFormCompTextLEN = len(noFormCompText)

          perCnoF = noFormCompTextLEN / origTextLEN
          perCisF = formCompTextLEN / origTextLEN

          print('\n')
          print('# of characters in original version :  ' + str(origTextLEN))
          print('# of characters in formatted compressed version :  ' + str(formCompTextLEN)
          )
          print('# of characters in NON-formatted compressed version :  ' + str(noFormCompTe
          xtLEN))
          print('compression ratio of formatted compressed version :  ' + str(perCisF))
          print('compression ratio of NON-formatted compressed version :  ' + str(perCnoF))
          print('total runtime :  ' + str(tTot) + ' sec')
          print('DONE DONE DONE !!!')
          print('\n')
```

```
# of characters in original version :   402665
# of characters in formatted compressed version :   540258
# of characters in NON-formatted compressed version :   393457
compression ratio of formatted compressed version :   1.341705884544224
compression ratio of NON-formatted compressed version :   0.9771323556802801
total runtime :   2.377018928527832 sec
DONE DONE DONE !!!
```

Alternate Version

And then compress *Tom Sawyer* again, this time using the alternate version of the routine.

```
In [14]:  t0 = time.time()
          compTextArrAa = lzComprA(textInA)
          t1 = time.time()
```

To get the results

```
In [15]: formCompText = compTextArrAa[2]
         noFormCompText = compTextArrAa[1]

         tTot = t1 - t0
         origTextLEN = len(textInA)
         formCompTextLEN = len(formCompText)
         noFormCompTextLEN = len(noFormCompText)

         perCnoF = noFormCompTextLEN / origTextLEN
         perCisF = formCompTextLEN / origTextLEN

         print('\n')
         print('# of characters in original version :  ' + str(origTextLEN))
         print('# of characters in formatted compressed version :  ' + str(formCompTextLEN)
         )
         print('# of characters in NON-formatted compressed version :  ' + str(noFormCompTe
         xtLEN))
         print('compression ratio of formatted compressed version :  ' + str(perCisF))
         print('compression ratio of NON-formatted compressed version :  ' + str(perCnoF))
         print('total runtime :  ' + str(tTot) + ' sec')
         print('DONE DONE DONE !!!')
         print('\n')
```

```
# of characters in original version :  402665
# of characters in formatted compressed version :  2
# of characters in NON-formatted compressed version :  2
compression ratio of formatted compressed version :  4.9669079756124815e-06
compression ratio of NON-formatted compressed version :  4.9669079756124815e-06
total runtime :  0.1863260269165039 sec
DONE DONE DONE !!!
```

King James Version of the Bible

Now, we can run and time the compression routine for *King James Version of the Bible*.

Initial Verion

By using the first routine we defined

```
In [16]: t0 = time.time()
         compTextArrB = lzCompr(textInB)
         t1 = time.time()
```

To get the results

In [17]:
```python
formCompText = compTextArrB[2]
noFormCompText = compTextArrB[1]

tTot = t1 - t0
origTextLEN = len(textInB)
formCompTextLEN = len(formCompText)
noFormCompTextLEN = len(noFormCompText)

perCnoF = noFormCompTextLEN / origTextLEN
perCisF = formCompTextLEN / origTextLEN

print('\n')
print('# of characters in original version :  ' + str(origTextLEN))
print('# of characters in formatted compressed version :  ' + str(formCompTextLEN)
)
print('# of characters in NON-formatted compressed version :  ' + str(noFormCompTe
xtLEN))
print('compression ratio of formatted compressed version :  ' + str(perCisF))
print('compression ratio of NON-formatted compressed version :  ' + str(perCnoF))
print('total runtime :  ' + str(tTot) + ' sec')
print('DONE DONE DONE !!!')
print('\n')
```

```
# of characters in original version :  4351875
# of characters in formatted compressed version :  4598217
# of characters in NON-formatted compressed version :  3499160
compression ratio of formatted compressed version :  1.0566059457130548
compression ratio of NON-formatted compressed version :  0.8040580209679736
total runtime :  352.93959760665894 sec
DONE DONE DONE !!!
```

Alternate Version

And then compress *The King James Version of the Bible* again, this time using the alternate version of the routine.

In [18]:
```python
t0 = time.time()
compTextArrBa = lzComprA(textInB)
t1 = time.time()
```

To get the results

In [19]:
```python
formCompText = compTextArrBa[2]
noFormCompText = compTextArrBa[1]

tTot = t1 - t0
origTextLEN = len(textInB)
formCompTextLEN = len(formCompText)
noFormCompTextLEN = len(noFormCompText)

perCnoF = noFormCompTextLEN / origTextLEN
perCisF = formCompTextLEN / origTextLEN

print('\n')
print('# of characters in original version :  ' + str(origTextLEN))
print('# of characters in formatted compressed version :  ' + str(formCompTextLEN)
)
print('# of characters in NON-formatted compressed version :  ' + str(noFormCompTe
xtLEN))
print('compression ratio of formatted compressed version :  ' + str(perCisF))
print('compression ratio of NON-formatted compressed version :  ' + str(perCnoF))
print('total runtime :  ' + str(tTot) + ' sec')
print('DONE DONE DONE !!!')
print('\n')
```

```
# of characters in original version :  4351875
# of characters in formatted compressed version :  2
# of characters in NON-formatted compressed version :  2
compression ratio of formatted compressed version :  4.595720235530662e-07
compression ratio of NON-formatted compressed version :  4.595720235530662e-07
total runtime :  2.0846071243286133 sec
DONE DONE DONE !!!
```

With the compression part done, we move on to LZ Decompression

LZ-Decompression

We will first define a method for decompression a compressed| text file using the Lempel-Ziv Compression routine specified in the book.

```
In [ ]:  def lzDEcompr(inArray):
             mySize = len(inArray)
             mySize1 = mySize - 1

             myEndSize = len(inArray[mySize - 1])

             deCompText = ""

             for i in range(mySize1):
                 row = inArray[i]

                 numNow = row[0]
                 charNow = row[1]

                 if (numNow == 0):
                     deCompText += charNow
                 else:
                     wordNow = charNow

                     while (numNow != 0):
                         newRow = inArray[numNow]

                         charNow = newRow[1]
                         wordNow = charNow + wordNow

                         numNow = newRow[0]

                     deCompText += wordNow


             row = inArray[mySize1]
             if (myEndSize == 1):
                 tmpNum = row[0]

                 rowNow = inArray[tmpNum]

                 charNow = rowNow[1]
                 numNow = rowNow[0]

                 if (numNow == 0):
                     deCompText += charNow
                 else:
                     wordNow = charNow

                     while (numNow != 0):
                         newRowNow = inArray[numNow]

                         charNow = newRowNow[1]
                         wordNow = charNow + wordNow

                         numNow = newRowNow[0]

                     deCompText += wordNow

             else:
                 charNow = row[1]
                 numNow = row[0]

                 if (numNow == 0):
                     deCompText += charNow
                 else:
                     wordNow = charNow

                     while (numNow != 0):
                         newRowNow = inArray[numNow]
```

Run on Compressed Tom Sawyer

First, we run the decompression routine on the compressed *Tom Sawyer*.

```
In [ ]:  t0 = time.time()
         tomDecomp = lzDEcompr(compTextArrAa)
         t1 = time.time()
```

Then compare the lengths

```
In [ ]:  print(len(textInA))
         print(len(tomDecomp))
```

and the texts themselves

```
In [ ]:  tomDecomp == textInA
```

and, finally, and elapsed time

```
In [ ]:  tTOT = t1 - t0
         print(tTOT)
```

Run on Compressed King James Version of the Bible

First, we run the decompression routine on the compressed *King James Version of the Bible*.

```
In [ ]:  t0 = time.time()
         bibleDecomp = lzDEcompr(compTextArrBa)
         t1 = time.time()
```

Then compare the lengths

```
In [ ]:  print(len(textInB))
         print(len(bibleDecomp))
```

and the texts themselves

```
In [ ]:  textInB == bibleDecomp
```

and, finally, and elapsed time

```
In [ ]:  tTOT = t1 - t0
         print(tTOT)
```