Colin Keenan

ECE 5200

2-25-2020

ECE 5200 Exercise 1

**Date:** 2/4/20, Thursday, week 2

**Due Date:** 2/25/20 Thursday, week 5

**Total Points:**  45 points

Type the solutions in word document and submit to es-egr-03=>ECE52001=>Homework & Exam folder => Exercise 1. Type your C / C++ / Java / C# / MATLAB programming solutions in the corresponding file format and paste your output at the end of your programs..

## Analysis Question

1. (8%) Run the steganography web page of Andrew Tanenbaum as mentioned in the class (his web page is <http://www.cs.vu.nl/~ast/>)

You will get 5 Shakespeare plays in plain texts including Hamlet, Julius Caesar etc.

1. What is the size of the zebras.bmp file? Notice the author also mentions that the image is 1024 by 768 pixels where each pixel consists of three 8-bit numbers (in page 664 of Tanenbaum). Does this calculation coincide with the file size of zebras.bmp? What are the use of the three 8-bit numbers?

The original size of the zebras.bmp file is 2,359,350 bytes according to Windows. According to calculation the size of the file should be 1024 x 768 x 3 bytes = 2,359,296 bytes, which is 54 bytes smaller than the actual file size. Each 8-bit number corresponds to an 8-bit channel of color, RGB.

1. The last bit of each 8-bit number is used to encode (so when you get 8 bits they become an ASCII character or a byte). What is the number n of bytes in 1024 x 768 x 3 bits? Why we use 3 in the multiplication?

1024 x 768 x 3 bytes = 2,359,296 bytes, 3 is used because each pixel has 3, 8-bit channels one for Red, one for Green, and one for Blue color channels.

1. Add up the size of the 5 plays and the notes. Are they smaller or equal to the number n as calculated in (b)? If the size is bigger than n, what should be done?

The plays in size total to 201,788 + 115,805 + 176,952 + 103,609 + 120,927 + 15,810 = 734,891 bytes which is less than the total size of the image, so it can be successfully hidden within the image. If the size is larger, then the text files must be compressed, or the image needs to be larger.

2. (12%) For the Hill cipher you learned on pages 46 – 49 (5th edition), let’s try variations of the key matrix K of 2 x 2 dimension with different numbers.

Suppose the plaintext is “paymoremoney” as in the book.

Try these following variations of K:

1. K1 = 
2. K2 = 

Encrypt using these matrices and try to calculate the inverse of K if it exists (you need the inverse to decipher).

Encryption:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P | A | Y | M | O | R | E | M | O | N | E | Y |
| 15 | 0 | 24 | 12 | 14 | 17 | 4 | 12 | 14 | 13 | 4 | 24 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

1. Using K1 =  \* Text as 2D matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| = E I | = G A | = B L | = S Y | = P R | = C G |

PAYMOREMONEY = EIGABLSYPRCG

1. Using K2 =  \* Text as 2D matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| = X E | = C G | = J B | = G S | = F P | = S C |

PAYMOREMONEY = XECGJBGSFPSC

Inverse Calculation:

Neither K matrices can be inverted because the determinant of K1 is -2 mod 26 = 24, so 24 \* x = 1 mod 26 to make K1 invertible, which would mean than an even number times any number mod 26 becomes 1, which is impossible. Also the GCD of 24 and 26 is 2, not one. With K2, it is a similar story, with the determinant where the determinant is 13, which means that 13 \* x = 1 mod 26, which is impossible because since 13 is half of 26, 13 times any number in modulo 26 arithmetic would either be 0 or 13, not 1, therefore K2’s inversion is also impossible. Furthermore, their GCD is 13, not one, meaning they are not relatively prime to each other.

### Programming Questions

1. (15%) Write a computer program (in any computer language you are familiar with) to do *frequency analysis* of the appearance of all the 26 characters in the 5 different Shakespeare plays. Tabulate the results by hand or display them by program. How do the results of the 5 plays look? Are they close or far apart?

Code:

import string, csv  
import matplotlib.pyplot **as** plt  
  
def characterOccurrence(filename):  
 characterIndex = dict.fromkeys(string.ascii\_lowercase, 0)  
 with **open**(filename) **as** fileobj:  
 **for** line in fileobj:   
 **for** ch in line:  
 **if** ch in characterIndex:  
 characterIndex[ch] += 1  
 **return** characterIndex  
  
def dictionary2csv(dictionary, filename):  
 **w** = csv.writer(**open**(filename, "w"))  
 **w**.writerow([filename])  
 **for** key, val in dictionary.items():  
 **w**.writerow([key, val])  
  
def dictionary2plot(dictionary, name):  
 plt.figure(name)  
 plt.bar(range(len(dictionary)), **list**(dictionary.values()), align='center')  
 plt.xticks(range(len(dictionary)), **list**(dictionary.keys()))  
 plt.title(name)  
 plt.show()  
  
hamlet = characterOccurrence("Hamlet.txt")  
julius\_caesar = characterOccurrence("Julius\_Caesar.txt")  
king\_lear = characterOccurrence("King\_Lear.txt")  
macbeth = characterOccurrence("Macbeth.txt")  
merchant = characterOccurrence("Merchant.txt")  
  
# dictionary2csv(hamlet, "hamlet.csv")  
# dictionary2csv(julius\_caesar, "julius\_caesar.csv")  
# dictionary2csv(king\_lear, "king\_lear.csv")  
# dictionary2csv(macbeth, "macbeth.csv")  
# dictionary2csv(merchant, "merchant.csv")  
  
# dictionary2plot(hamlet,"Hamlet Character Occurrence")  
# dictionary2plot(julius\_caesar,"Julius Caesar Character Occurrence")  
# dictionary2plot(king\_lear,"King Lear Character Occurrence")  
# dictionary2plot(macbeth,"Macbeth Character Occurrence")  
# dictionary2plot(merchant,"Merchant Character Occurrence")  
  
plt.figure("Character Occurrences")  
plt.subplot(3,2,1)  
plt.bar(range(len(hamlet)), **list**(hamlet.values()), align='center')  
plt.xticks(range(len(hamlet)), **list**(hamlet.keys()))  
plt.title("Hamlet")  
plt.subplot(3,2,2)  
plt.bar(range(len(julius\_caesar)), **list**(julius\_caesar.values()), align='center')  
plt.xticks(range(len(julius\_caesar)), **list**(julius\_caesar.keys()))  
plt.title("Julius Caesar")  
plt.subplot(3,2,3)  
plt.bar(range(len(king\_lear)), **list**(king\_lear.values()), align='center')  
plt.xticks(range(len(king\_lear)), **list**(king\_lear.keys()))  
plt.title("King Lear")  
plt.subplot(3,2,4)  
plt.bar(range(len(macbeth)), **list**(macbeth.values()), align='center')  
plt.xticks(range(len(macbeth)), **list**(macbeth.keys()))  
plt.title("Macbeth")  
plt.subplot(3,2,5)  
plt.bar(range(len(merchant)), **list**(merchant.values()), align='center')  
plt.xticks(range(len(merchant)), **list**(merchant.keys()))  
plt.title("Merchant")  
plt.show()

Plots:

A screenshot of a cell phone

Description automatically generated

As you can see from the plots above, each of the plays character distributions hold a very similar structure small variations which is to be expected, but the overall structure and “shape” of the character layouts are the same.

1. (10%) Implement the credit card validation algorithm (Luhn’s modulo 10 formula that you can find on internet easily) and valid numbers (for example, all visa cards start with 4, master cards start with 51-55, etc.) covered in the class. Refer to cardtype.pdf in the class folder if necessary, write a C / C++ / C# / Java / MATLAB program that validates credit numbers.

Provide an example of a valid credit card number and an invalid number as input for your program. Turn in the output as well (note, in order not to disclose your credit number, you may either alter your card number or *make* a valid number by the rule). I will test your program using my credit card number(s).

Code:

**def** **validator**(n):  
 validatelist=[]  
  
 **for** i **in** n:  
 validatelist.append(int(i))  
  
 **for** i **in** range(0,len(n),2):  
 validatelist[i] = validatelist[i]\*2  
 **if** validatelist[i] >= 10:  
 validatelist[i] = validatelist[i]//10 + validatelist[i]%10  
  
 **if** sum(validatelist)%10 == 0:  
 print('This a valid credit card')   
 **else**:  
 print('This is not valid credit card')  
  
**def** **cardnumber**():  
 result=''  
 **while** True:  
 **try**:  
 result = input('Please enter the 16 digit credit card number : ')  
 **if** **not** (len(result) == 16) **or** **not** type(int(result) == int) :  
 **raise** Exception  
 **except** Exception:   
 print('That is not a proper credit card number. \nMake sure you are entering digits not characters and all the 16 digits.')  
 **continue**  
 **else**:  
 **break**  
  
 **return** result  
  
**def** **goAgain**():  
 **return** input('Do you want to check again? (Yes/No) : ').lower()[0] == 'y'  
  
**while** True:  
 result = cardnumber()  
 validator(result)  
  
 **if** **not** goAgain():  
 **break**

Output:



~~Extra Credit (8%)~~

~~(8%) Security: Consider this question from Tanenbaum. This question asks you to break the cryptic text as follows (the text is a poem by Lewis Caroll, who wrote )~~

~~kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm~~

~~ur mfudm zhx mftnm zhx mdzythc pzq ur ezsszcdm zhx gthcm~~

~~ur zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm~~

~~sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk~~

~~The translation of the first word is:~~

*~~the~~*

1. **~~Find~~** ~~the mapping of the characters (character ‘k’ to ‘t’, ‘f’ to ‘h’ ‘d’ to ‘e’ since “kfd” maps to “the” etc.) as many as possible from the first sentence. Show how derive this in detail. Do~~ **~~not~~** ~~just state the mapping (you may have guessed that).~~
2. ~~How many characters have been mapped in the first sentence?~~
3. ~~Some words in the cryptic text are not translated by the partial information of the fully translated first sentence. For example, pzq in the second sentence could be translated to wa? since ‘p’ to ‘w’ and ‘z’ to ‘a’ from the first sentence. Can you use your English knowledge to complete the word? Some educated guess and trial error may be necessary.~~
4. ~~Complete the translation of all 4 sentences using your reasoning.~~
5. ~~What do you have to say about this so called substitution cipher?~~