ECS 40 **Program #7 (50 points)** **Winter 2016**

Due : Wednesday, March 2nd, 11:59pm in p7 of cs40a.
New concepts: string, iterators, sort, and vector.
Filenames: authors.csv, Makefile-Debug.mk, decipher.cpp

Netbeans (10 points)

Netbeans is installed on the CSIF computers under Applications->Programming->Netbeans, or just Searching for Netbeans. You can download your own free copy from netbeans.org. From the Download page choose the "C/C++" version. There are many tutorials available from the Netbeans start page or directly at http://www.netbeans.org/kb/trails/cnd.html. It is up to you to choose what tutorials you need.

To match the indentation requirement of 2 spaces, within NetBeans you will first need to create a C++ project. Once you are working with a C++ project, go to Tools->Options->Editor->Formatting and set the "Number of Spaces per Indent" and "Tab Size" to 2. The "Expand Tabs to Spaces" should already be checked.

To prove that you used NetBeans, we want you to handin Makefile-Debug.mk for your decipher project. This is worth 10 points of the 50 points for the assignment. To have Makefile-Debug.mk created properly, you must create a C/C++ Application Project in Netbeans (don't select C/C++ Project with Existing Sources). Makefile-Debug.mk is located in the nbproject directory of the directory NetBeans will create for your decipher project. Make sure it's there after creating your project. Even though NetBeans creates its own Makefile for your project, we will be compiling your program by entering: g++ -Wall -ansi decipher.cpp. You should make sure that your program also compiles without warnings using that command!!!

Decipher (40 points, 1.25 hours)

You are to write a program that can read in file (given as a command line parameter) that encrypts its alphabetic letters using a Caesar cipher, and then outputs the six most likely deciphered messages from the 26 possible messages. For this program we will use the following definition for likelihood:

$Likelihood = \sum_{i='A'}^{i<='Z'} f(i) * e(i)$, where *f(i)* is the frequency of a letter in normal English text, and *e(i)* is the frequency of the letter in the encrypted message. I created a class with two ints, and an operator< that was useful for this.

You will find the file named frequencies in ~ssdavis/40/p7 that contains the frequencies of the letters in a normal English text of 1000 characters. You will also find my version of decipher.out there too. The format of your output must match mine. I found isupper() and islower() of <cctype> and sort() and count() of <algorithm> useful.

5 points Extra Credit: Use the dictionary in ~ssdavis/40/p7/words to select only decrypted messages that contain only English words in them. If you do the extra credit, you should explore all 26 possibilities. If you do the extra credit, then you may have a total of three strings. See the example runs at the end of this assignment.

Further specs:
1. You may not use any C arrays, nor "[]" anywhere in your program other than argv[1]. This means will be using vectors, at(), and iterators throughout the program.
2. You may only declare one string for the entire program, nor may you have a vector<char>. You may not have a vector of strings.
3. You may not use the c_str() function of the string class.
4. Fill a single string with the entire contents of the file using a stringstream. Do a search of the web to find the solution.

The Caesar Cipher
From http://www.cs.trincoll.edu/~crypto/historical/caesar.html

"One of the simplest examples of a substitution cipher is the *Caesar cipher*, which is said to have been used by Julius Caesar to communicate with his army. Caesar is considered to be one of the first persons to have ever employed encryption for the sake of securing messages. Caesar decided that shifting each letter in the message would be his standard algorithm, and so he informed all of his generals of his decision, and was then able to send them secured messages. Using the Caesar Shift (3 to the right), the message,
`"RETURN TO ROME"` would be encrypted as,
`"UHWXUA WR URPH"`

In this example, 'R' is shifted to 'U', 'E' is shifted to 'H', and so on. Now, even if the enemy did intercept the message, it would be useless, since only Caesar's generals could read it. Thus, the Caesar cipher is a *shift* cipher since the ciphertext alphabet is derived from the plaintext alphabet by shifting each letter a certain number of spaces. For example, if we use a shift of 19, then we get the following pair of ciphertext and plaintext alphabets:

```
Plaintext:  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
Ciphertext: T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
```

To encipher a message, we perform a simple substitution by looking up each of the message's letters in the top row and writing down the corresponding letter from the bottom row. For example, here is an enciphered message:

```
THE FAULT, DEAR BRUTUS, LIES NOT IN OUR STARS BUT IN OURSELVES
MAX YTNEM, WXTK UKNMNL, EBXL GHM BG HNK LMTKL UNM BG HNKLXEOXL.
```

Essentially, each letter of the alphabet has been shifted nineteen places ahead in the alphabet, wrapping around the end if necessary. Notice that punctuation and blanks are not enciphered but are copied over as themselves.

Breaking a Caesar Cipher by Hand

Can a computer guess what shift was used in creating a Caesar cipher? The answer, of course, is yes. But how does it work? The unknown shift is one of 26 possible shifts. One technique might be to try each of the 26 possible shifts and check which of these resulted in readable English text. But this approach has limitations. For one thing how would the computer recognize "readable English text?" For another, what if a multiple Caesar shift was used, as is the case for a Vigenere cipher , where each letter of the keyword provides the basis for a Caesar shift. That is, if the key word is bam, then every third letter of the plaintext starting at the first would be shifted by 'b' (=1) and every third letter beginning at the second would be shifted by 'a' (=0) and every third letter beginning at the third would be shifted by 'm' (=12). Obviously we can't depend on obtaining readable English text here.

A better approach makes use of statistical data about English letter frequencies. It is known that in a text of 1000 letters of various sources, the English alphabet occur with about the following relative frequencies:

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
73	9	30	44	130	28	16	35	74	2	3	35	25	78	74	27	3	77	63	93	27	13	16	5	19	1

This information can be useful in deciding the most likely shift used on a given enciphered message. Suppose the enciphered message is:

```
K DKVO DYVN LI KX SNSYD, PEVV YP CYEXN KXN PEBI, CSQXSPISXQ XYDRSXQ.
```

We can tally the frequencies of the letters in this enciphered message, thus

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	4	3	0	0	0	3	0	4	1	0	4	1	4	3	1	6	0	0	4	0	7	5	0

Now we can now shift the two tallies so that the large and small frequencies from each frequency distribution match up roughly. For example, if we try a shift of ten on the previous example, we get the following correspondence between English language frequencies and the letter frequencies in the message.

English Language Frequencies

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
73	9	30	44	130	28	16	35	74	2	3	35	25	78	74	27	3	77	63	93	27	13	16	5	19	1

Enciphered Message Frequencies

K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
4	1	0	4	1	4	3	1	6	0	0	4	0	7	5	0	0	1	2	4	3	0	0	0	3	0

Note that in this case the large frequencies for cipher X and Y correspond to large for English N and O, the bare spots for cipher T and U correspond to bare spots for English J and K. Also, an isolated large frequency for cipher S corresponds to a similar one for English I. In view of this evidence we needn't even worry too much about the drastic mismatch for English E, which is usually the most frequent letter in a random sample of English text. If we now apply this substitution to the message we get:

```
A TALE TOLD BY AN IDIOT, FULL OF SOUND AND FURY, SIGNIFIYING NOTHING."
```

```
.
[ssdavis@lect1 p7]$ cat File6.txt
A TALE TOLD BY AN IDIOT, FULL OF SOUND AND FURY, SIGNIFYING NOTHING.
 [ssdavis@lect1 p7]$ encipher.out
Please enter a file name>> File6.txt
Please enter a start letter (a-z) >> k
[ssdavis@lect1 p7]$ cat File6.txt.enc
K DKVO DYVN LI KX SNSYD, PEVV YP CYEXN KXN PEBI, CSQXSPISXQ XYDRSXQ.
[ssdavis@lect1 p7]$
[ssdavis@lect1 p7]$ decipher.out File6.txt.enc
Deciphered message version #1
A TALE TOLD BY AN IDIOT, FULL OF SOUND AND FURY, SIGNIFYING NOTHING.

Deciphered message version #2
G ZGRK ZURJ HE GT OJOUZ, LARR UL YUATJ GTJ LAXE, YOMTOLEOTM TUZNOTM.

Deciphered message version #3
Z SZKD SNKC AX ZM HCHNS, ETKK NE RNTMC ZMC ETQX, RHFMHEXHMF MNSGHMF.

Deciphered message version #4
L ELWP EZWO MJ LY TOTZE, QFWW ZQ DZFYO LYO QFCJ, DTRYTQJTYR YZESTYR.

Deciphered message version #5
N GNYR GBYQ OL NA VQVBG, SHYY BS FBHAQ NAQ SHEL, FVTAVSLVAT ABGUVAT.

Deciphered message version #6
F YFQJ YTQI GD FS NINTY, KZQQ TK XTZSI FSI KZWD, XNLSNKDNSL STYMNSL.

[ssdavis@lect1 p7]$ cat File2.txt.enc
Eayq otmzsqe mdq ea exai, kag pa zaf zafuoq ftqy.
Aftqde mdq ea rmef, ftqk pa zaf zafuoq kag.
[ssdavis@lect1 p7]$ decipher.out File2.txt.enc
Deciphered message version #1
Some changes are so slow, you do not notice them.
Others are so fast, they do not notice you.

Deciphered message version #2
Iecu sxqdwui qhu ie ibem, oek te dej dejysu jxuc.
Ejxuhi qhu ie vqij, jxuo te dej dejysu oek.

Deciphered message version #3
Hdbt rwpcvth pgt hd hadl, ndj sd cdi cdixrt iwtb.
Diwtgh pgt hd uphi, iwtn sd cdi cdixrt ndj.

Deciphered message version #4
Eayq otmzsqe mdq ea exai, kag pa zaf zafuoq ftqy.
Aftqde mdq ea rmef, ftqk pa zaf zafuoq kag.

Deciphered message version #5
Rnld bgzmfdr zqd rn rknv, xnt cn mns mnshbd sgdl.
Nsgdqr zqd rn ezrs, sgdx cn mns mnshbd xnt.

Deciphered message version #6
Wsqi glerkiw evi ws wpsa, csy hs rsx rsxmgi xliq.
Sxlivw evi ws jewx, xlic hs rsx rsxmgi csy.

[ssdavis@lect1 p7]$
```

```
[ssdavis@lect1 p7]$ encipher.out
Please enter a file name>> File5.txt
Please enter a start letter (a-z) >> n
[ssdavis@lect1 p7]$ decipher.out File5.txt.enc
Deciphered message version #1
Ted Ex

Deciphered message version #2
Its Tm

Deciphered message version #3
Epo Pi

Deciphered message version #4
Don Oh

Deciphered message version #5
Alk Le

Deciphered message version #6
Hsr Sl

[ssdavis@lect1 p7]$ decipherXC.out File5.txt.enc
Deciphered message version #1
Ted Ex

Deciphered message version #4
Don Oh

Deciphered message version #7
Paz At

[ssdavis@lect1 p7]$ encipher.out
Please enter a file name>> File8.txt
Please enter a start letter (a-z) >> p
[ssdavis@lect1 p7]$ decipher.out File8.txt.enc
Deciphered message version #1
bog core cork cot cozy curl cut hall ham harry has hat hay he hom hot
 hue huey hug huh hum hush ingot jam jot joy jut lax layout lorry lot
 lotte lutz mae marin marry mat may mu mug mull nan nat nest no
null orr ott out oz pam pare paz punt put rot rout rue rush sally
 sam satin sod sorry tarry vat wag wage yuh

Deciphered message version #2
uhz vhkx vhkd vhm vhsr vnke vnm atee atf atkkr atl atm atr ax ahf ahm
 anx anxr anz ana anf anla bgzhm ctf chm chr cnm etq etrhnm ehkkr ehm
 ehmmx enms ftx ftkbg ftkkr ftm ftr fn fnz fnee gtg gtm gxlm gh
gnee hkk hmm hnm hs itf itkx its ingm inm khm khnm knx knla lteer
 ltf ltmbg lhw lhkkr mtkkr otm ptz ptzx rna

Deciphered message version #3
via wily wile win wits wolf won buff bug bulls bum bun bus by big bin
 boy boys boa bob bog bomb chain dug din dis don fur fusion fills fin
 finny font guy gulch gulls gun gus go goa goff huh hun hymn hi
hoff ill inn ion it jug july jut john jon lin lion loy lomb muffs
 mug munch mix mills nulls pun qua quay sob
```

```
Deciphered message version #4
obt pber pbex pbg pbml phey phg unyy unz uneel unf ung unl ur ubz ubg
 uhr uhrl uht uhu uhz uhfu vatbg wnz wbg wbl whg ynk ynlbhg ybeel ybg
 ybggr yhgm znr zneva zneel zng znl zh zht zhyy ana ang arfg ab
ahyy bee bgg bhg bm cnz cner cnm chag chg ebg ebhg ehr ehfu fnyyl
 fnz fngva fbq fbeel gneel ing jnt jntr lhu

Deciphered message version #5
hum iuxk iuxq iuz iufe iaxr iaz ngrr ngs ngxxe ngy ngz nge nk nus nuz
 nak nake nam nan nas nayn otmuz pgs puz pue paz rgd rgeuaz ruxxe ruz
 ruzzk razf sgk sgxot sgxxe sgz sge sa sam sarr tgt tgz tkyz tu
tarr uxx uzz uaz uf vgs vgxk vgf vatz vaz xuz xuaz xak xayn ygrre
 ygs ygzot yuj yuxxe zgxxe bgz cgm cgmk ean

Deciphered message version #6
fsk gsvi gsvo gsx gsdc gyvp gyx lepp leq levvc lew lex lec li lsq lsx
 lyi lyic lyk lyl lyq lywl mrksx neq nsx nsc nyx peb pecsyx psvvc psx
 psxxi pyxd qei qevmr qevvc qex qec qy qyk qypp rer rex riwx rs
rypp svv sxx syx sd teq tevi ted tyrx tyx vsx vsyx vyi vywl weppc
 weq wexmr wsh wsvvc xevvc zex aek aeki cyl

[ssdavis@lect1 p7]$ decipherXC.out File8.txt.enc
Deciphered message version #1
bog core cork cot cozy curl cut hall ham harry has hat hay he hom hot
 hue huey hug huh hum hush ingot jam jot joy jut lax layout lorry lot
 lotte lutz mae marin marry mat may mu mug mull nan nat nest no
null orr ott out oz pam pare paz punt put rot rout rue rush sally
 sam satin sod sorry tarry vat wag wage yuh

Deciphered message version #3
via wily wile win wits wolf won buff bug bulls bum bun bus by big bin
 boy boys boa bob bog bomb chain dug din dis don fur fusion fills fin
 finny font guy gulch gulls gun gus go goa goff huh hun hymn hi
hoff ill inn ion it jug july jut john jon lin lion loy lomb muffs
 mug munch mix mills nulls pun qua quay sob

[ssdavis@lect1 p7]$
[ssdavis@lect1 p7]$ decipherXC.out File7.txt
Deciphered message version #2
I Pitt

Deciphered message version #7
A Hall

Deciphered message version #19
U Buff

[ssdavis@lect1 p7]$
```