**Computer Science 1106 Programming Assignment One --- 50 Points**

The usage of user-defined functions in these programs is optional.

**Exercise 1:**

**Telephone Number Generator**

Standard telephone keypads contain the digits 0 through 9. The numbers 2 through 9 each have 3 letters associated with them, as is indicated by the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| Digit | Letter | Digit | Letter |
| 2 | ABC | 6 | MNO |
| 3 | DEF | 7 | PQRS |
| 4 | GHI | 8 | TUV |
| 5 | JKL | 9 | WXYZ |

Companies use seven letter words as a memory mnemonic for consumers of their products. For example, a seven letter word like NUMBERS would translate to 686-2377. Unfortunately, your company recently upgraded their telephone system and it cannot handle words for telephone numbers. They need a program to translate their old phone business directory which consists of seven letter words into their numeric representation.

Write a program to convert the existing phone directory into their new phone directory. That is, convert each seven letter word into its numeric equivalent based upon the chart above. The file to use for input is oldPhone.txt and the new file should be called newPhone.txt.

**Exercise 2:**

Write a C++ program that reads a file and counts the number of alphabetic characters, number of uppercase letters, number of punctuation characters, and the number of whitespace characters in the file. Utilize the get function to process the file one character at a time. The data file to use is the Gettysburg Address, which is found below. Your program should utilize the **cctype** functions found [here](http://www.cplusplus.com/reference/clibrary/cctype/). The program should produce a report containing the above information.

Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we cannot dedicate -- we cannot consecrate -- we cannot hallow -- this ground. The brave men, living and dead, who struggled here, have consecrated it, far above our poor power to add or detract. The world will little note, nor long remember what we say here, but it can never forget what they did here. It is for us the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us -- that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion -- that we here highly resolve that these dead shall not have died in vain -- that this nation, under God, shall have a new birth of freedom -- and that government of the people, by the people, for the people, shall not perish from the earth.

**Due Date:**

* As indicated on the drop box folder: Programming Assignment One

**What to turn in:**

1. For each exercise copy and paste your source code into a word document and paste screen shots of the execution of your program in the directly under your source code in the document.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// Programmer: Chris Dang Class: CSCI 1106 Fall 2014

//

// Description: Program takes 7 letter words from a text file and will turn

// them into numbers that are associated under a 10 digit phone numpad.

// They will then be outputted into a file.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Chart: Comes from standard 10 digit numbpad on phones

Letter: Digit:

ABC 2

DEF 3

GHI 4

JKL 5

MNO 6

PQRS 7

TUV 8

WXYZ 9

\*/

#include <iostream>

#include <fstream> // Needed for in-file / out-file

#include <cctype> // Needed for toupper

#include <string>

using namespace std;

int main () {

char letter;

int number;

string oldNumber;

// Grabbing text file with phone numbers

ifstream infile;

infile.open("oldPhone.txt");

if(!infile){

cerr << "Error opening file. --- Terminating Program. " << endl <<

endl;

exit (1106);

} // end if

// Outputting the new number file

ofstream outfile;

outfile.open("newPhone.txt");

if(!outfile){

cerr << "Error writing file. --- Terminating Program. " << endl <<

endl;

exit (1106);

} // end if

while(infile >> oldNumber){ //Grabs info from file

for (int i = 0; i < 7; i++) { //counter loop for outputting '-' and endl

//loop runs on values 0 to 6. There are 7 values for the 7 digits

//in the phone number.

letter = toupper(oldNumber.at(i)); //simplifies the input for the

// switch

switch (letter){

case 'A':

case 'B':

case 'C': number = 2;

break;

case 'D':

case 'E':

case 'F': number = 3;

break;

case 'G':

case 'H':

case 'I': number = 4;

break;

case 'J':

case 'K':

case 'L': number = 5;

break;

case 'M':

case 'N':

case 'O': number = 6;

break;

case 'P':

case 'Q':

case 'R':

case 'S': number = 7;

break;

case 'T':

case 'U':

case 'V': number = 8;

break;

default:

number = 9;

break;

}// end switch

outfile << number; // outputs number by number to the file

if (i == 2) // if the third value is outfiled, add '-'

outfile << '-';

if (i == 6) // if the last (7th) value is outfiled, add endl

outfile << endl;

}// end for

} // end while;

cout << "Output file has been processed." << endl << endl ;

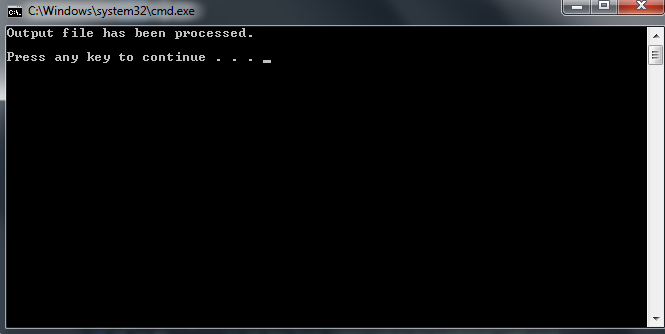
// closes the in file and out file.

infile.close();

outfile.close();

return 0;

}

****

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//**

**// Programmer: Chris Dang Class: CSCI 1106 Fall 2014**

**//**

**// Description: Program reads in Gettysburg Address file. It will then process**

**// the file and display the number of alpabetical characters, number of**

**// uppercase characters, number of punctuation marks, and number of white**

**// spaces.**

**//**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**#include <iostream>**

**#include <cctype> // Needed for character processing**

**#include <fstream> // Needed for in-file / out-file**

**#include <string>**

**using namespace std;**

**int main () {**

**char character;**

**int alpCount = 0, uppCount = 0, pncCount = 0, spcCount = 0;**

**// alphabetical, uppercase, punctuation, white space**

**// Opening text file with Gettysburg Address**

**ifstream infile;**

**infile.open("gettysburgAddress.txt");**

**if(!infile){**

**cerr << "Error opening file. --- Terminating Program. " << endl << endl;**

**exit (1106);**

**} // end if**

**while (infile.get(character)) { //grabs character from file**

**if (isalpha(character)){ //bracketing for efficiency; isupper**

**alpCount++ ; //only runs if isalpha runs because**

**if (isupper(character))//if !alpha, then character cannot**

**uppCount++ ; // be an upper character.**

**}**

**if (ispunct(character))**

**pncCount++ ;**

**if (isspace(character))**

**spcCount++ ;**

**} // end while**

**cout << " The number of alphabet characters is: " << alpCount << endl;**

**cout << " The number of uppercase characters is: " << uppCount << endl;**

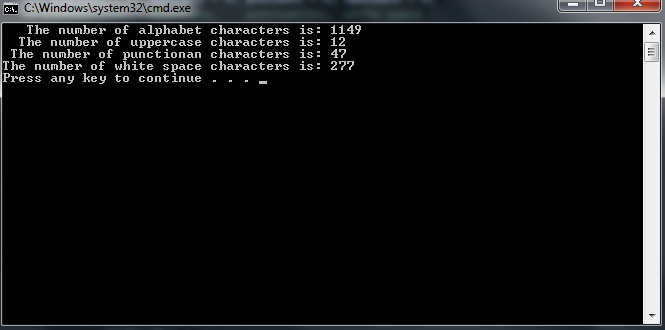
**cout << " The number of punctionan characters is: " << pncCount << endl;**

**cout << "The number of white space characters is: " << spcCount << endl;**

**infile.close();**

**return 0;**

**}**

****

1. The file containing the phone numbers from exercise one.
2. A hard copies of the source code for each program.
3. Compress all files into **a single file** with the filename: yournameProgram1.zip
4. Place the compressed file into the D2L drop box called Programming Assignment One.
5. Hand in the hard copies of the programs and the word document at the beginning of class on the due date indicated by the folder in the D2L drop box.