COMP1602: Computer Programming II
Lab #6

1. (a) Write a function, *matchingLetters*, which accepts two C-strings representing a pair of words as parameters and determines how many letters in corresponding positions of each word are the same, regardless of case. For example, if the two words are *coat* and *CATTLE*, they have two (2) matching letters, 'c' at location 0 and 'T' at location 3. The function has the following heading:

```
int matchingLetters (char word1[], char word2[]);
```

   (b) Write a *main* function which accepts two words from the user at the keyboard and finds the number of matching letters in the two words.

2. (a) An *ious* word is a word which **ends** with the letters *ious*. For example, prev*ious* and stud*ious* are *ious* words. Write a function, *isIousWord*, which accepts a C-string *str* as a parameter and returns *true* if *str* is an *ious* word and *false*, otherwise. The function has the following heading:

```
bool isIousWord (char str[]);
```

   (b) Write a *main* function which accepts a word from the user at the keyboard and determines if it is an *ious* word.

3. A C-string, *str***,** contains a sequence of lowercase letters of the alphabet. Write a program which accepts *str* from the user at the keyboard and find the *length* of the *longest sequence of identical letters* in *str*. The program should also display the letter in the longest sequence. For example, if *str* is "aabbc**dddddd**eeebbbaaaacccd", your program should display that the length of the longest sequence is 6, formed by the letter 'd'.

4. (a) Write a function, *doubleLetter*, which accepts a C-string *str* and a character *c* as parameters and returns *true* if there are two consecutive occurrences of the character *c* in *str*, and *false*, otherwise. For example, *doubleLetter ("hello", 'l')* returns *true* and *doubleLetter ("cattle", 'c')* returns *false*. The function has the following heading:

```
bool doubleLetter (char str[], char c);
```

   (b) Write a program which accepts a C-string, *s*, from the user at the keyboard and a character *t*, and determines if there are two consecutive occurrences of *t* in *s*.

5. (a) Write a function, *removeNonLetters* which accepts a C-string *str* as a parameter and removes all the characters that are not letters from *str*. The function has the following heading:

```
void removeNonLetters (char str[]);
```

   (b) Write a program which accepts a C-string from the user at the keyboard and removes all the characters that are not letters from the C-string.

6. *This question is similar to Question 4 from Lab #5 but the program has more features.*

   Each line of a text file, *words.txt*, contains a word or a phrase. The last line of the file contains the word "END". The word/s in a line are separated by spaces and may contain punctuation and other characters. The first two lines of data in the file are as follows:

   ```
   Madam
   Madam, in Eden, I'm Adam.
   ```

   Write a *main* function which reads the word/s from each line, removes the characters that are not letters and determines if the word/s in each line form a palindrome. The output from the program should start as follows:

   ```
   Word/Phrase                          Is Palindrome?

   Madam                                Yes
   Madam, in Eden, I'm Adam.            Yes
   ```

   You will need to use the solution for Question 4 of Lab #5 with the *removeNonLetters* function from Question 5. Also, if *inputFile* is the input file stream, use *inputFile.getline(word, 80)* to read all the characters of a line into the *word* C-string (declared as "char word [80]").

7. Two words or phrases are anagrams if they contain the same letters rearranged, regardless of case and punctuation characters. For example, "Meteor" and "remote" are anagrams. Also, "Election Results" and "Lies, Let's Recount!" are anagrams.

   (a) Write a function, *isAnagram*, which accepts two C-strings, *str1* and *str2*, as parameters and returns *true* if *str1* and *str2* are anagrams and *false*, otherwise.

   (b) The lines of a text file, *wordpairs.txt*, are organized in pairs. Each pair of lines contains a word or phrase, separated by spaces and may contain punctuation and other characters. The last line of the file contains the word "END". The first four lines of data in the file are as follows:

   ```
   Meteor
   remote
   Election Results
   Lies, Let's Recount!
   ```

   Write a program which reads the words or phrases in each pair of lines and determines if the words/phrases are anagrams. Your program must use the *removeNonLetters* function from Question 5 to remove the characters that are not letters from each line. Given the data in *wordpairs.txt*, the first two lines of output should be as follows:

   ```
   Word/Phrase 1            Word/Phrase 2            Are Anagrams?

   Meteor                   remote                   Yes
   Election Results         Lies, Let's Recount!     Yes
   ```

   NB: If *inputFile* is the input file stream, use *inputFile.getline(str, 80)* to read all the characters of a line into the *str* C-string (declared as "char str [80]"). You will also need to use functions from Lab #5 such as the *indexOf* function (Question 5), the *upperToLower* and *isEqual* functions (Question 3) as well as functions such as *length* and *isLetter*.