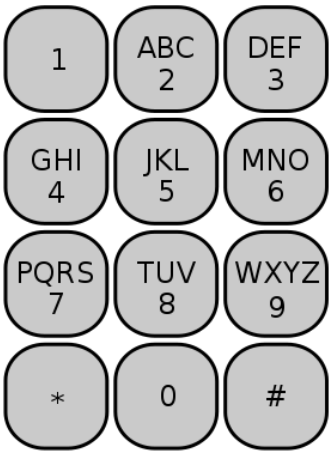
EGR125 Due date: _____
Introduction to Engineering Methods (C++)
File: N125P4B

Programming Assignment #4: Text Messaging

Background:

Text messaging on cell phones is sometimes configured so that when the user types a sequence of number keys, possible words appear. The numbers on the keypad correspond to the following letters:

2 – abc
3 – def
4 – ghi
5 – jkl
6 – mno
7 – pqrs
8 – tuv
9 – wxyz



For example, if the user enters the sequence 368253 the word "double" should appear.
However, if the user enters the sequence 4663, several possible words could be formed, including "home", "good", "gone", "hood", "hone", "hoof", and "goof." The user might be given a way to select from this list of potential words. To keep this assignment from becoming too complicated, we will avoid punctuation and special symbols.

US Dictionary:

A text file named USDictionary.txt is provided on the instructor's web page. This file contains over 118,000 words representing the English language. This dictionary is not arranged like most dictionaries, but instead lists words as follows:

- All 1-letter words (in alphabetical order)
- All 2-letter words (in alphabetical order)
- All 3-letter words (in alphabetical order)
- Etc

Program Description

Write a C++ program that begins by giving the user three options:

1) Enter a word and convert it to a sequence of numbers (this should be easy!)
   Example: User input: Buffalo
   Output: 2833256

2) Enter a sequence of numbers and display a list of all possible words in USDictionary.txt that match. Note that since the longest word in USDictionary.txt is 45 letters in length, the input number sequence must be a string (since no integers in C++ can handle 45 digits).
   Example 1: User input: 368253
   Output: double
   – no parentheses are needed if there is only one word that matches

   Example 2: User input: 4663
   Output: (good, gone, hood, hone, hoof, good)
   – use parentheses and separate the possible words by commas if two or more words match

3) Read a message from a text file (a sequence of numbers separated by white spaces) and display the message. Give the user the choice of one of the 7 sample messages found on the instructor's web site (plus two of your own messages).

   Example 1:
   User Input: User selects Message 4 which contains 4 6333 2 639 7932874478
   Output: I need a new sweatshirt

   Example 2:
   User Input: User selects Message 5 which contains 4 268448 2 8733 87825
   Output: I bought a (used, tree) (truck, usual)

The program must also satisfy a number of specific requirements as listed below.

Program Requirements:

1) Write and use **class KeyPad** as described below:
   - Data members: Include data members for
     - Number (a string consisting of a series of numbers)
     - Word (a string consisting of a series of letters)
   - Member functions: Include member functions for
     - Reading Numbers from the keyboard and checking if they are valid (2-9)
     - Reading Words from the keyboard, converting to lowercase, and checking if they are valid (lowercase a-z)
     - Converting Words to Numbers
     - Converting Numbers to Words and displaying all possible words from USDiscitionary.txt
     - You can add additional member functions if you wish
   - Non-member functions: Include non-member functions for
     - Converting words to all lowercase letters
     - You can add additional non-member functions if you wish.
   - Main function:
     - The main function should use class KeyPad and should give the user the three options as described in the Program Description section.
     - The user should be able to enter words with any combination of uppercase and lowercase letters.
     - In addition to the 7 messages on the instructor's web page (Message1.txt – Message7.txt), add two messages of your own. Each message should have 6 or more words and should form a meaningful sentence. One message should have only one possible solution. The other message should have multiple solutions.
     - Give an error message if a word is specified that is not in the dictionary
     - Give the user the option of re-running the program
   - File structure: Use separate header and implementation files for class KeyPad.

Output:

Turn in printouts for the following cases:

Option 1) Convert at least 5 words to numbers, including the words "baseball", "engineer", and three words of your choice containing at least 5 letters

Option 2) Convert the following numbers into words (listing all possible words)
2429253
52925
776337767
4663
9355
5668
4653

Option 3) Convert all 9 messages to sentences (7 messages on the instructor's web page and two of your own as previously described).

Extra Credit Suggestions: (for a maximum of 10 additional points)

1. If a word is not in the dictionary, give the user the option of adding it to a personal dictionary. Rather than modifying USDictionary.txt, create a new dictionary (PersonalDictionary.txt) that words can be added to each time the program is run. Be sure to search both dictionaries for valid words.
2. Add the use of punctuation in messages to the program.
3. Use your imagination!