Problem Statement
Computers have long been used to encode and decode secret messages. One of the simplest "encryption" schemes is Pig Latin. There are various rules for translating English to Pig Latin. For this assignment, we use the following rules:

1. If a word starts with a vowel (not including 'y') and ends in a consonant, simply append "ay" to the end of the word. Example: *insect* becomes *insectay*.
2. If a word starts and ends with a vowel (not including 'y'), simply append "hay" to the end of the word. Example: *else* becomes *elsehay*.
3. Otherwise, remove the consonants up to the first vowel of the word, append them to the end of the word, and then append "ay". Example: *clog* becomes *ogclay*.
4. All spacing and punctuation is retained.
5. Capitalization should be retained. Example: *Deborah Hwang* becomes *Eborahday Anghway*.

Assignment
Write a Pig Latin translator that will repeatedly read lines of text from the console and translate them until the user enters "QUIT". You may assume that each line of original or translated text will be no more than 1000 characters. It is highly recommended that the following functions be written and in the order listed. Please note that while these exact functions are not required, programs that do not make use of functions as appropriate will not earn full credit for analysis and design.

- A function `is_vowel` that receives a character and returns true if the character is a vowel, and returns false otherwise.
- A function `get_next_word` that receives an English sentence as a string and a start index, and passes back the next word (i.e., consecutive alphabetic characters) that starts at the start index in the sentence as a (separate) string. The start index is assumed to be the index of a character that is a letter.
- A function `get_next_nonword` that receives an English sentence as a string and a start index, and passes back a (separate) string of all the non-alphabetic characters that occur starting at the start index. The start index is assumed to be the index of a character that is not a letter.
- A function `translate_word` that receives an original English word as a string and passes back the corresponding Pig Latin word translation as a (separate) string.
- A function `translate` that receives an original English sentence as a string and passes back the corresponding Pig Latin sentence translation as a (separate) string. The main idea of this function is to alternate calling `get_next_word` and `get_next_nonword` keeping track of the starting index of each word/non-word until reaching the end of the original English sentence string. A word is translated before concatenating it to the result Pig Latin sentence string, while a non-word is concatenated to the result string as is.

It is recommended that each function above be tested for correctness before going on to the next function. The final main program is responsible for repeatedly reading in a line of English text, calling the `translate` function, and displaying the resulting Pig Latin text.
The output of the program must conform to the following example (user input in **bold**).

Igpay Atinlay Anslatortray

Please enter a sentence to translate (QUIT to exit):
**The quick brown fox jumped over the lazy dog.**

The translation is:
Ethay uickqay ownyray oxfay umpedjay overay ethay azylay ogday.

Please enter a sentence to translate (QUIT to exit):
**A man, a plan, a canal, Panama. Palindrome!**

The translation is:
Ahay anmay, ahay anplay, ahay analcay, Anamay. Alindromepay!

Please enter a sentence to translate (QUIT to exit):
**The University of Evansville pig latin translator ... is online.**

The translation is:
Ethay Universityay ofay Evwisevillehay igpay atinlay anslatortray ... isay onlinehay.

Please enter a sentence to translate (QUIT to exit):
**Wile E. Coyote, Super-Genius**

The translation is:
Ileway Ehay. Oyotecay, Upersay-Eniusgay

Please enter a sentence to translate (QUIT to exit):
**QUIT**

**Coding Notes**
- There is a newline (that causes a blank line) before the prompt for input and before the output.
- There is a newline at the end of the prompt for input. It should be written **immediately** after the colon. I.e., there is no space after the prompt.
- The line of input must be read in using **fgets**, since the input may have spaces. Reminder: **fgets** stores the newline from when the user presses the Enter key.
- Reminder: make sure the constructed strings end with the null-terminator ('\0').

**REMINDER:** Your program must compile for it to be graded. Submissions that do not compile will be returned for resubmission and assessed a late penalty. Submissions that do not substantially work also will be returned for resubmission and assessed a late penalty.

Follow the program documentation guidelines in the C Programming Style Guideline handout. As stated in the syllabus, part of the grade on a programming assignment depends on how well you adhere to the guidelines. The grader will look at your code and grade it according to the guidelines.

**What to Submit**
Electronically submit a zipfile containing **main.c** (only) as explained in the handout Submission Instructions for CS 210. The submission system will start accepting assignments by the evening of Sunday, March 20.