PigLatin Assignment for C Programming

M. Beeson

March 9, 2010

1 The Assignment

This assignment involves practice with strcpy, strcat, strtok, and malloc or strdup, as well as practice writing a for-loop and using break and continue to control it.

It involves converting a string given in ordinary English (“plaintext”) into a simple coded form often used by children. This form is called “Pig Latin”. (Wikipedia says the origins of Pig Latin are unknown.) Most American and English children know what Pig Latin is, but students from other countries may need it explained, and even Americans will need the rules set out with enough precision for a programmer.

First we have to distinguish between vowels and consonants. The vowels are a, e, i, o, and u, and of course their upper-case versions as well. In ordinary English, sometimes y counts as a vowel, but for this assignment it does not. The consonants are all letters that are not vowels. The “initial consonants” of a word are the series of consecutive consonants with which it begins. Often that is just one consonant, as in “dog”, but it can also be two consonants, as in “church”, or even three, as in “scram”. You may assume that there will not be more than ten initial consonants.

The basic rule for converting a word to Pig Latin is to remove the initial consonants, add them on again at the end of the word, followed by the fixed suffix ay. For example, dog becomes ogday, and church becomes urchchay. To convert a sentence to Pig Latin, you convert the words one by one, leaving punctuation alone.

For this assignment you have to implement two functions:

```c
void pigLatinWord(const char *x, char *ans)
/* translate one word (no spaces) into pig Latin and write the
result into ans, assuming ans has enough space to hold the result.
Rules for pig Latin are given in the assignment, including rules
about capitalization.
If *x ends in a punctuation mark, that ending
punctuation mark is placed after the added suffix; for example
"torpedoes," becomes "orpedoestay,". Punctuation marks are
defined as semicolon, comma, period, question mark, and exclamation point.
You may assume that *x contains only letters and numbers,
except possibly for a single punctuation mark at the end. You may
assume that it does not have more than ten initial consonants. You may
not assume anything about the length of x. (It could arbitrarily long.)
You are not allowed to alter the characters of x.
*/
```
void pigLatinSentence(const char *x, char *ans)
/* x will be a null-terminated string. Separate the string x into words at spaces,
and translate each word into pig Latin,
and write the result (space-separated) into ans.
Assumes that ans has enough space to hold the answer.
Example: if x is "See Sally run", then ans should be ‘‘Eesay Allysay unray’’.
See the assignment for the precise rules about capitalization and punctuation.
The function is not allowed to alter the characters of x.
It must free any memory that it allocates using malloc or calloc.
It must not assume any fixed maximum on the length of the string x,
or any fixed maximum on the number of words in x, or any fixed
maximum on the length of an individual word.
If multiple spaces occur between words, they are condensed to one space in the output.
For example, "dog days" becomes "ogday aysday".
*/

The grading system will award you a C if pigLatinWord passes all tests. If you want a higher
grade than C then pigLatinSentence must also work correctly. In any case you have to submit some
code that will compile at least for pigLatinSentence.
The required filename is PigLatin.c. You should, as usual, prepare a header file PigLatin.h
and a file main.c for your own use, but submit only PigLatin.c

2 The detailed specification of Pig Latin

In addition to the rules given above, the following rules will apply:
1. Should the suffix be AY or ay?
   If the original word is all in uppercase use AY, else use ay, unless the original word consists of
   a single character. For example I becomes Iay, not IAY; but IRS become IRSAY and CHURCH
   becomes URCHCHAY.
2. What happens to initial capitals?
   If a word is not all in uppercase, then an initial capital becomes lower case, but the new initial
   letter is capitalized instead. For example, Dog becomes Ogday, not ogDay or ogday or OgDay. As
   another example, Now becomes Ownay. Note that punctuation is not counted in determining if a
   word is all caps. Thus DOG; becomes OGDAY; even though it ends in a semicolon.
3. Punctuation is not changed. For example,
   Damn the torpedoes, full speed ahead.
   becomes
   Amnday ethay orpedoestay, ullfay eedspay aheaday.

3 Programming Hints for pigLatinWord

1. Start by writing a main() program to test your work. You should do this before writing any
   other code. See the last section below for hints, but finish reading this section first.
2. Declare an array of, say, 32 characters (anyway at least eleven) to hold the initial consonants.
   Copy the initial consonants into that array and add a null terminator.
3. Determine if the input consists of all caps or not.
4. Copy the rest of the input into the ans array. Be sure to capitalize the first letter if necessary.
5. If x ends in a punctuation mark, save that punctuation mark and overwrite it in ans with a null character.
6. Add the initial consonants at the end of ans. Then add the correct suffix (either AY or ay).
7. If there was a punctuation mark, add it at the end, paying proper attention to the null terminator.

4 Programming Hints for pigLatinSentence

1. Start by writing a main() program to test your work. You should do this before writing any other code. See the next section for hints, but finish reading this section first.
2. We are not allowed to modify the input string, but we would like to use strtok on it to identify the words. So, we need to make a copy of it. This can be done with malloc and strcpy, or with strdup. Please read the documentation of strdup.
3. Write a for-loop. The body of the for-loop should use strtok to get the next word. Words are space-separated. You will have to read the documentation of strtok carefully. Once you get the next word, if it is blank (which can happen if there are multiple spaces) skip it. (To do that, use continue.) If it is not blank, call your pigLatinWord function to convert it to Pig Latin, writing it at the end of what is currently in ans. Unless this is the very first word written into ans, you will need to first write a space character ' ' to separate this word from the previous one. Note that to write "dog" into the ans array starting at index 5, you can use strcpy(ans+5,"dog");
4. Do you need to free the string returned by strtok, or not? Read the documentation to find out.
5. When there are no more words ("tokens"), that will be indicated by the return value of strtok. Read the documentation to find out how. Then use break to exit your loop. Since you are going to exit the loop this way, the middle part of the for statement just contains 1, rather than specifying any fixed upper bound on the number of iterations of the loop. You don’t know how many words there might be.
6. Before exiting your function, you must free the memory that was allocated to hold the copy of the input string. Otherwise you will have a “memory leak”.

5 Testing your Program

Please write a very thorough main() program to test your code. Do not just test it on one example at a time. I will get you started with this code:

```c
static char *examples[] = {
};

};
```
Write a loop that calls your `pigLatinWord` function on each example, and compares the result to the stored answer. Use the `strcmp` function to compare two strings. Note that it returns 0 if the strings are equal. You cannot use `x==y` to see if two strings have the same characters. That only compares whether the addresses of the first character are the same. If the two strings don’t compare correctly, print something out.

Once `pigLatinWord` is working correctly, run similar tests on `pigLatinSentence`. Do not think that the few tests included here test every single point in the assignment!

Finally, check that you have indeed freed all memory you allocated. Your best bet in this small program is just to verify that by looking at the code.