CSC 111 Assignment 6

NOTE: Your programs must compile and execute using the Eclipse environment in ECS 242. If you do your work on your own computer, be sure to test it in ECS 242 before you submit it.

Programming instructions

Assignment 6 consists of two parts, both covering file I/O and string manipulation. In Part I, your code will read a text file—*The Adventures of Sherlock Holmes*—encode each word by randomly permuting the letters, and then write the encoded text to a file. Part II involves reading raw character data. This assignment deals with text files—a stream of characters. Text files are often organized into a sequence of lines. Each line is terminated by `'\n'`. The last line of a file may not contain a trailing `'\n'` character.

Part I

The following text has circulated on the web since 2003.

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a word are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mind deos not raed ervey lteter by istlef, but the wrod as a wlohe.

According to a researcher (sic) at Cambridge University, it doesn't matter in what order the letters in a word are, the only important thing is that the first and last letter be at the right place. The rest can be a total mess and you can still read it without problem. This is because the human mind does not read every letter by itself, but the word as a whole.

This text is attributed to the MRC Cognition and Brain Sciences Unit, Cambridge University, UK. Graham Rawlinson, University of Nottingham, UK, wrote a dissertation on the subject entitled "The Significance of Letter Position in Word Recognition." [www.mrc-cbu.cam.ac.uk/people/matt.davis/Cmabrigde].

In Part I you are to encode text as illustrated above: the letters of each word, except the first and last letters, will be scrambled. The assignment is to read a text file—*The Adventures of Sherlock Holmes*—encode each word and then write the scrambled text to an output file.

Instructions

- Download the file `A6P1_2017_TestingSherlock.txt` and `A6P1_2017_Sherlock.txt` and store it in the directory where you develop your program.
- Download the template `A6P1_2017_Template.c` for this part of Assignment 6.
- Read the input file by reading one line at a time (using `fgets()`) into a string buffer and then separating each line into individual words using `sscanf()`.
- For each word, scramble the order of the interior letters of the word (leaving the first and last letters intact). You may use any scheme to reorder letters, as long as it scrambles all words of four

or more letters. One possible scheme is to use random numbers to swap random indices of the string representing the word, or to use a left- or right-shift algorithm to shift the elements of the string by one or more positions. In the event that the word contains punctuation (e.g. '.' or ';'), treat the punctuation characters as letters.

- After scrambling each word, write the result to an output file, with a maximum of 50 words per line. You are not required to exactly replicate the spacing in the input file, but you must insert at least one space between words.
- The primary input file, `A6P1_2017_Sherlock.txt`, is too long for testing purposes; use the shortened `A6P1_2017_TestingSherlock.txt` file (or create an even shorter text file yourself). Note that for marking, your code will be tested with different input files.

Part II

The `getchar()` function in the `stdio.h` library reads a single character from the user and returns it (as a value of type `char`). The `ctype.h` header file contains several useful functions for working with char values, including the following:

- `isspace(c)` – Returns `true` if the character c is any kind of space (including a newline or a tab character) and `false` otherwise.
- `isalpha(c)` – Returns `true` if the character c is a letter (uppercase or lowercase) and `false` otherwise.
- `isdigit(c)` – Returns `true` if the character c is a digit (0 through 9) and `false` otherwise.
- `islower(c)` – Returns `true` if the character c is a lowercase letter and `false` otherwise.
- `isupper(c)` – Returns `true` if the character c is an uppercase letter and `false` otherwise.
- `tolower(c)` – Returns the lowercase version of the provided character c.
- `toupper(c)` – Returns the uppercase version of the provided character c.

A syntactically correct C program has been provided in the file `A6P2Template.c` (posted on CourseSpaces). As written, the template program reads a single line of text from the user and prints each character on a line by itself. The result of a sample run of the unmodified template is shown below.

```
Enter a line of text: Hello World
Character: H
Character: e
Character: l
Character: l
Character: o
Character: (space)
Character: W
Character: o
Character: r
Character: l
Character: d
```

Your assignment is to modify the provided template to read a line of text from the user and print out the text (in one line) with the **first letter of each word capitalized and multiple blanks between words reduced to one blank**. It should be possible to complete the assignment by modifying only the contents of the while loop in the `read_and_capitalize()` function in the template. The output of several sample runs of a model solution is given below:

Sample Run 1:

```
Enter a line of text: west virginia
West Virginia
```

Sample Run 2:

```
Enter a line of text: blue ridge mountains
Blue Ridge Mountains
```

Sample Run 3:

```
Enter a line of text: shenandoah river
Shenandoah River
```

Assignment submission instructions

CSC 111 assignments will only be accepted electronically through the assignment page on the CSC 111 CourseSpaces site. Your submission will consist of ***two C source files*** named by the following convention: If your student ID is ***V00123456,*** your C source files for parts 1 and 2 (respectively) must be named **V00123456A6P1.c and V00123456A6P2.c**. In addition, your ***full name, student ID,*** and ***Assignment name*** (e.g., Assignment 6) must appear in a ***comment section at the beginning of each C program***.

For example (in Part 1) – not required for the HTML file:

```
Name: Polar Bear (Replace this with your name)
UVicID: V00123456 (Replace this with your student number)
Date: 2017/11/04 (Replace this with the date you wrote the program)
Assignment: A6
File name: V00123456A6P1.c (Replace V00123456 with your student number)
Description: This program generates scrambled Sherlock Holmes
```

Please submit only the source file and not the executable file. To verify that you have submitted the correct file, you are strongly encouraged to download your submissions from the site and test that they work correctly in your Eclipse environment. Submissions that do not follow the guidelines above will receive a mark of zero.

Since this assignment only requires three source files, CourseSpaces will only allow you to submit three files. However, until the assignment due date, you may change your submission by deleting and resubmitting your source files multiple times. After the due date, no submissions will be accepted. When grading is complete, your assignment mark and comments will appear in the ***Gradebook*** section of CourseSpaces.