CSC 362 Programming Assignment #1
Due Date: Friday, February 1

This assignment will test you over inputting and outputting files character-by-character, and the use of logic. We have a text file that has been written in a bizarre code. The code works like this: the message consists of alphabetic characters that alternate from upper to lower and back to upper case. Any other letters that appear in the message that are not in this sequence are "dummy characters" to be discarded. In addition, the message can contain non-alphabetic characters which should be output as is. The upper-to-lower-to-upper sequence restarts with each new sentence (that is, after any period, we will assume no other character ends a sentence such as ! or ?). The only upper case letters that should be output in upper case are the first letter of any sentence. All other letters, no matter if they appear in upper or lower case in the input, should be lower cased. Consider the following text:

```
ABCbcdCCCdefghEE AfafGHIJKhij IIII jklmKLMlmnMNO nonoOP
pqrQQQ.  abcSTtttUUv abcWXYxxx YYZ.
```

Start with the 'A' which should be output as 'A'. We next expect a lower case letter so 'B' and 'C' are discarded. Next is 'b' which is output as 'b'. Now, we expect an upper case letter so 'c' and 'd' are discarded. The first 'C' is the next upper case letter but since it is not starting a new sentence it is output as 'c'. Since we expect a lower case letter, the following two 'C's are discarded followed by 'd' which is output as 'd'. Now we expect an upper case letter so 'e', 'f', 'g' and 'h' are discarded. The first 'E' is output as 'e' and the second 'E' is discarded. The space is output as is. The next word's 'A' is discarded as we are still expecting a lower case letter. 'f' is output as 'f' and the 'a' and 'f' that follow it are discarded. The 'G' is output as 'g'. This continues until the '.' is reached, which is output as '.' and we reset to start looking for an upper case letter. So, 'a', 'b' and 'c' are discarded and 'S' is output as 'S'. The output of the two lines above would be:

```
Abcde fgh i jklm no
pq.  Stuv wx y.
```

All of the text output of the decoded message should be sent, character-by-character, to a text output file. Your program should also output to the console (using printf) a summary of the processing of the file specifying: the input file's name, the output file's name, the number of characters in the input file, the number of characters in the output file (these counts include all non-alphabetic characters such as periods, spaces, new lines, etc), and the output file's size compared to the input as a percentage (computed as output size / input size * 100). The percentage should be formatted appropriately with at least 1 decimal point of accuracy and the % sign. For instance, in the above case, there are 92 input characters (80 letters, 2 periods, and 9 spaces, and one \n at the end of the first line. NOTE: there are two spaces between the first period and the 'a', there is one space not visible at the end of the first line. There are 36 output characters. The percentage should be 39.13% (depending on how many decimal points of accuracy you decide to provide). Remember to properly cast because this is an integer division. Do not use float/double variables to count the number of characters as these numbers are int values. The console output might look like this:

```
Input file:                     input1.txt
Output file:                    output1.txt
Number of input characters:     92
Number of output characters:    36
Percent reduction in output:    39.13%
```

NOTE: your input and output sizes may differ by 1-2 characters because of EOF characters inserted by Windows.

From ctype.h, you can use isalpha, islower, isupper, tolower, toupper. DO NOT USE ARRAYS for any part of this assignment except to store file names. You can input the filenames from the user using scanf or hardcode them in your program like: `char name[]="input1.txt";` There are three input files on the website for you to test. The first, input1.txt, is the same as the text above. After you have debugged your program, run it on input2.txt and input3.txt. Hand in print outs of your *commented* code, the two output files created and the two console windows obtained when running on the input files input2.txt and input3.txt.