

Intro to Engineering Computing
Spring 2020
Homework #3
Due March 6, 2020 @ 5:00 pm

Braille is a system of “touch” reading and writing that uses patterns of raised dots to convey letters, numbers and symbols. It is a useful way for individuals with visual impairments and blindness to read and write, but interestingly, the ideas behind the writing system originated with Napoleon’s army. In the early 1800s, a French soldier, Charles Barbier, witnessed fellow soldiers getting killed at night because they were using lamps to read messages. Mr. Barbier conceived of a system of “night writing” that would allow soldiers to safely read messages without lamplight, and the idea of raised-letter communication was born. Later, Louis Braille took inspiration from Barbier’s night writing technique, and developed what we now know as Braille writing, in which a 2x3 array of dots are used to represent letters and numbers, based on which of the six dots are raised.

Braille is a classic example of a binary code - each of the six dots in the Braille “cell” can either be “off” (0 or *not raised*) or “on” (1 or *raised*). Count the six dots of the Braille cell as from 1-3 (left column) and 4-6 (right column). Expressed as binary a value, the letter “A,” [in which dot #1 (upper left) is *raised* (or “on”) and dots 2-6 are *not raised* (or “off”)] can be written 100000. Similarly, the letter “H,” [in which dots #1, 2, and 5 are “on” and dots 3, 4, and 6 are “off”] would be written 110010.

The Braille Cell

1	●	●	4
2	●	●	5
3	●	●	6

<div><div>●</div><div>○</div><div>○</div><div>○</div><div>○</div><div>○</div></div>	A	100000	<div><div>●</div><div>○</div><div>●</div><div>●</div><div>○</div><div>○</div></div>	H	110010
---	---	--------	---	---	--------

Our IEC Braille message in file ‘HW3_Braille.txt’ uses this six-digit binary expression of Braille letters for its message. White spaces are used to separate letters, and words are separated by 999999. Note that for simplicity, we are going to ignore lowercase/capital letter differences, and as well as punctuation.

Here are your tasks for this homework:

1. Open the file named ‘HW3_Braille.txt’ for reading.
 - a. Convert the Braille writing in that file into a message using the standard English alphabet. (Note that Braille is not a separate language, but a way of writing languages – the message expressed in Braille would still be in “English” and hence the term “translate” is not used in this context).
 - b. Display the converted, standard English message on the console.
 - c. Write the same converted message to a file named ‘message.txt’. After writing the file, close the file. The contents of this file should be in standard English.

2. Now open the file 'message.txt' for reading.
 - a. Read the standard English text from the file and convert each letter in the text into your binary Braille "code."
 - b. Using the same convention as the initial file (spaces separating letters, 999999 separating words), write the binary Braille, letter by letter, to a new file called 'message2.txt' and also write the same to the console.
 - c. When the last letter in the text has been converted and written in binary Braille, close the output file.
3. To ensure your program ran correctly, open 'message2.txt' and 'HW3_Braille.txt' and, by eye, check that the contents are exactly the same.
4. Submit your source code (*.cpp) to the class's ICON site dropbox by 5:00PM, March 6, 2020.

Hints:

- Recall from the textbook (pp. 76-79), that `cin` can be used as `cin.get(ch1)` to get a single character from the input and store it in the character variable called `ch1`. You can use the `.get()` function with an `ifstream` variable to read a single **character** from a file in the same manner. Note that if you are reading in an integer, this would not be the approach to take, since you would only get a single digit of the integer each time you called the `.get()` function. Instead, for integers, use the `>>` operation as we did in **Lecture 9**.
- Be careful when you are comparing your 6-digit input integer with specific values – does the computer read in `010000` as `010000` or as `10000`?
- Optionally: a file called "binary_Braille_conv.txt" is available for download and contains two columns of info – the first column has the 6-digit number and the second column contains the corresponding letter represented by the number. If you choose, your code can open and read this file to search for matching elements so as to decode/encode the message as needed.
- The library `iomanip` has useful functions that you may want to take advantage of when performing file input/output. Recall how we used some of these functions in **Lecture 8**, when creating a nicely-formatted Multiplication Table.



























GRADING Rubric (100 points total):

- 80 points for working code:
 - 40 points for correctly converting the Braille code in 'HW3_Braille.txt' into standard English letters and writing the results to the file 'message.txt' and to the console.
 - 5 points for checking that the file opened correctly.
 - 10 points for converting the Braille code in the 'HW3_Braille.txt' file into English.
 - 10 points for displaying the standard written English message on the console.
 - 10 points for writing the English message to a file named 'message.txt'.
 - 5 points for closing the 'message.txt' file.
 - 40 points for correctly converting the standard English text into binary Braille (as described on page 1) and writing the result to the file 'message2.txt' and to the console.
 - 5 points for checking that the file opened correctly.

- 10 points for converting the standard written English text in the 'message.txt' file into binary Braille.
- 10 points for displaying the binary Braille message on the console.
- 10 points for writing the binary Braille to a file named 'message2.txt'.
- 5 points for closing the 'message2.txt' file.
- 20 points for style (See the Style guide under Content on ICON):
 - 5 points for indenting -- #5 in the Style Guide
 - 5 points for in-line comments -- See #7 in the Style Guide
 - 5 points for comment blocks -- See #8 in the Style Guide
 - 5 points for meaningful variable names -- See #10 in the Style Guide

DO NOT WORK TOGETHER! Students caught working together on this assignment will **drop a whole letter grade** for the course!

NOTE: If your program does not compile on an ECS Linux machine or you do not submit a .cpp file by the deadline, you will receive a zero on your homework. Remember: late homework is not accepted.

Braille alphabet				File <i>binary_Braille_conv.txt</i>
 A	 B	 C	 D	100000 A
 E	 F	 G	 H	110000 B 100100 C 100110 D 100010 E 110100 F 110110 G 110010 H
 I	 J	 K	 L	010100 I 010110 J 101000 K 111000 L
 M	 N	 O	 P	101100 M 101110 N 101010 O 111100 P
 Q	 R	 S	 T	111110 Q 111010 R 011100 S 011110 T
 U	 V	 W	 X	101001 U 111001 V 010111 W 101101 X
 Y	 Z			101111 Y 101011 Z