CSCI 2226 and 6620 Data Structures

Program 1: Run Length Compression

1 Goals of this Assignment

1. To identify a compiler and an IDE that will meet your needs for this course.
2. To write a program in C++.
3. To become familiar with the file I/O system of C++.
4. Using an acceptable style: no goto, no global variables, proper indentation, reasonable names.

2 File Compression

Most people now use rar or zip to compress files. These applications use a combination of compression mechanisms, in sequence. This assignment explores the simplest compression scheme we have. Other assignments this term will explore a more complex compression scheme.

2.1 Run Length Compression

This algorithm is effective on text or binary files that have the same byte repeated over and over. Think of a file containing a table of numbers: it has lots of consecutive space characters, and may have a repeated filler character, such as a '.' .

In this scheme, any "run" of the same character (4 or more identical consecutive bytes) is replaced by a triplet of bytes, consisting of

1. An escape character. We will use 0x7f, which is sometimes called "esc". It is a non-printing ASCII character.
2. The letter that has been repeated.
3. A 1-byte count = the number of repetitions.

To make this work, any esc character, or run of them, that occurs in the input must *also* be replaced by a triplet: esc esc count .

2.2 Implementing the Algorithm.

1. At all times, keep a "prior char" variable to compare to the current input. Also maintain a runCount.
2. To begin, set the run count to 1 and read the first input char into your priorChar variable using a method that does NOT skip whitespace.
3. Now enter a processing loop.
   (a) Read an input character. (do not skip whitespace.) If it is the same as the prior character:
   - If the run counter has reached 255, write a triplet and reinitialize the counter to 1
   - Else increment the run counter.

   (b) If the input is different from the prior character:
   - If the count is 1, 2, or 3, write out the prior character 1, 2, or 3 times.
   - Else (the count is 4 or more) write out a triplet.

   In both cases, save the current input in prior.
4. When end of file occurs, write out the last char (prior) an appropriate number of times.

3 Expanding the Compressed File

Every compression scheme needs a matching decompression scheme. Decompression is done in a loop, thus:

1. Read an input character, ch (do not skip whitspace) and quit if end of file happens.
2. If ch is NOT an escape character, write it out and continue at the top of the loop.
3. If ch IS an escape character, read two more bytes: a letter and a count.
4. Write the letter out count times and continue at the top of the loop.

4 Due February 1, 2016

All programming in this course will be done in C++. This assignment involves file I/O which must be done using the C++ I/O libraries. Standards for style, organization, and OO usage will increase gradually throughout the term. For this program, do it simply. You do not need any classes or structs. You can do the whole thing in main().

Students who already know some C++ should write the compression program (section 2). This includes most of the CSCI 2226 students and one or two CSCI 6620 students.

Students who are just beginning to learn C++ should write the expansion program (Section 3). I will provide a compressed file for testing.

4.1 Submitting Your Work

Make sure your name is inside every file. Put your source files and your test output into a directory named P1-Smith (or whatever your name is). Do not include your entire project, just the source and output. Zip this directory. Hand in an electronic copy of your zipped directory by emailing it to my HOME email address: alice@we2skate.net