EE231002 Introduction to Programming

Lab13. String Encoding

Due: Dec. 30, 2017

Traditional English characters are encoded using 7-bit ASCII format. Thus, some existing network communication protocols assume 7-bit per byte streams. Other bytes longer than 7 bits are used for network control. For non-ASCII characters, such Chinese, these protocols have troubles handling their transmission. This limitation was recognized in the early days, and some encoding schemes were developed. In essence, these encoding schemes convert non-ASCII characters into ASCII streams making them transmittable using the standard protocols. Then, decoders are invoked to translate those streams back to the original format. In this way, non-ASCII streams can be easily transmitted using the existing framework.

In this assignment we will implement a special encoding scheme. A short encoded file is shown below:

```
begin
DP6E4B @@
end
```

The first line is a header. The encoded file always starts with the `begin` keyword. The encoded text starts from the next line. Each encoded line begins with a line lengths character, then followed by the encoded string. Depends on the original text, the number of encoded line varies and can be as large as needed. The tail of the encoded file is a single line, `end`.

The line length character is formed by treating this integer as a `char` and checking bit 5 of this `char`. If it is a 0, then bit 6 is set; otherwise, the `char` itself is the line length character.

The encoded string is formed by taking 3 original characters, which have total of 24 bits, split them into 4 groups, each group then has 6 bits. Two 0's are added to the left to form 8-bit `char` again. And then, bit 5 of each `char` is checked again. If a zero is found, bit 6 of the `char` is set. These 4 `char`'s are then the encoded text for the 3 original characters.

Original text:	C	a	t	
ASCII code:	0100,0011	0110,0001	0111,0100	
Splitting:	0001,0000	0011,0110	0000,0101	0011,0100
Check bit-5:	0101,0000	0011,0110	0100,0101	0011,0100
Encoded text:	P	6	E	4

The encoding process continues by taking the next 3 characters as input and producing 4 characters output. This process is repeated until all the input characters are processed. In the case that we have less than 3 characters left to be encoded. `NULL` characters (`'\0'`) are added to form 3 characters such that the preceding encoding process can be carried out. The line length

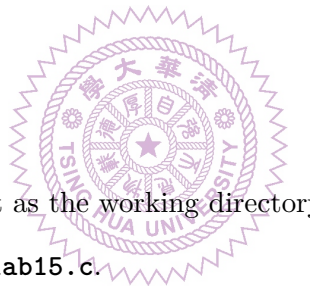
character, however, does not count these added `NULL` characters, and in decoding, they should not be printed out. Also note that all control characters, such as `\t`, `\n`, `\0`, are also encoded.

Your assignment is to write a `C` program to decode an encoded file. The program should read the header line and ignore it. Then convert the encoded string back to the original form by revering the process as shown in the figure above. Of course, it needs to detect the tail section and stop the conversion process. Two encoded files are provided for you to test your program: `etext.cd` and `ctext.cd`. You can use the standard Unix command line to redirect file as the standard input to the program as the following:

```
$ ./a.out < etext.cd
```

Of course, you can also save the decoded strings to a file using the following command as an example. By doing so, you can detect if any non-printable characters have been accidentally printed out.

```
$ ./a.out < etext.cd > etext
```



Notes.

1. Create a directory **`lab15`** and use it as the working directory.
2. Name your program source file as **`lab15.c`**.
3. The first few lines of your program should be comments as the following.

   ```
   /* EE231002 Lab13. String Encoding
      ID, Name
      Date:
   */
   ```

4. After you finish verifying your program, you can submit your source code by

   ```
   $ ~ee2310/bin/submit lab13 lab13.c
   ```

   If you see a "submitted successfully" message, then you are done. In case you want to check which file and at what time you submitted your labs, you can type in the following command:

   ```
   $ ~ee2310/bin/subrec lab13
   ```

   It will show the submission records of lab13.

5. You should try to write the program as efficient as possible. The format of your program should be compact and easy to understand. These are part of the grading criteria.