EE231002 Introduction to Programming

Lab13. Text Decoding

Due: Jan. 2, 2021

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 decoder for a special encoding scheme. A short encoded file is shown below:

```
58V]D90H@
0
```

An encoded file consists of a number of encoded lines. Each encoded line starts by a character indicating the number of original characters in that line. This leading character is formed by adding the number to the ASCII character `'0'`. In the example above, the first line consists of 5 original characters, and the second line contains 0 character, which also ends the encoded text. The maximum number of original characters in an encoded line is 45. In this case, the leading character is `]`.

The rest of a encoded line 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`. The ASCII character `' '` (space character) is then added to form a printable 7-bit ASCII character.

The encoding process for the example above is shown below.

Original text:	`c`	`o`	`d`	
ASCII code:	0110,0011	0110,1111	0110,0100	
Splitting:	0001,1000	0011,0110	0011,1101	0010,0100
Adding `' '`	0011,1000	0101,0110	0101,1101	0100,0100
Encoded text:	`8`	`V`	`]`	`D`
Original text:	`e`	`'\n'`	`' '`	
ASCII code:	0110,0101	0000,1010	0010,0000	
Splitting:	0001,1001	0001,0000	0010,1000	0010,0000
Adding `' '`	0011,1001	0011,0000	0100,1000	0100,0000
Encoded text:	`9`	`0`	`H`	`@`

The encoding process takes 3 original characters to form 4 encoded characters. In the example above, there are only 5 original characters. In such cases, where the number of original characters is not a multiple of 3, extra space characters `' '` are added to ensure the encoding process can be carried out.

Your assignment is to write a C program to decode an encoded file. Three encoded files are provided for you to test your program: they are `story1.cd`, `story2.cd` and `story3.cd`. You can use the standard Unix command line to redirect file as the standard input to the program as the following:

```
$ ./a.out < story1.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 < story1.cd > story1.txt
```

Notes.

1. Create a directory **`lab13`** and use it as the working directory.
2. Name your program source file as **`lab13.c`**.
3. The first few lines of your program should be comments as the following.

   ```
   // EE231002 Lab13. Text Decoding
   // 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.