Problem Set 2

Due before midnight on Monday, September 24, 2018.

This short assignment is designed to deepen your understanding of C++ I/O and of character representations.

1 Assignment Goals

1. Learn how to use command line arguments.
2. Learn how to open a file and read its contents.
3. Learn how characters are represented by bytes in the computer.
4. Learn the difference between a character and its ASCII code.
5. Learn how to obtain the ASCII code of a character stored in a variable of type `char`.
6. Learn how to print the character whose ASCII code is stored in a variable of type `int`.
7. Learn how to print an `int` as a decimal number.
8. Learn how to print an `int` as a hex number.
9. Learn how to test if a `char` is printable.
10. Learn how to use the output manipulators `dec`, `hex`, `setw()`, and `setfill()` to control the printed form of numbers.
11. Learn precisely what `in>>val` does to the istream `in` when `val` has type `int`.
12. Learn how to use `in.get(ch)` to read a single character from `in`.
13. Learn precisely what `out<<val` does to the ostream `out` when `val` has type `int` and when `val` has type `char`.
14. Learn how to recover after a failed attempt to read a decimal number from an istream.

2 Problem

You should review Lecture 4, which gives an overview of C++ I/O. You should also look at the material on C++ I/O in the two textbooks, Applied C and C++, Chapter 14 and Exploring C++, Chapter 3 to you make sure you understand I/O in enough depth to do this assignment. You can find details on the individual functions and manipulators in http://www.cplusplus.com/reference/iostream/ for.

You should write a program that takes the name of a file as a command line argument. The file will consist of a mixture of letters, digits, punctuation, whitespace characters, and control characters. Your program should open the file on an input stream `in` and declare an `int` variable `x`. It should then repeatedly attempt `in>>x`. If a number is successfully read into `x`, then `x` should be printed in decimal on a line by itself.

If the attempt to read `x` fails, then the next character should be read from the stream using `in.get(ch)`, where `ch` has type `char`, and a one-line "Skipping..." message should be printed. Depending on the character read, the message might look like either of the following:

```
Skipping char: 116 0x74 't'
Skipping char:   0 0x00
```

In each case, the ASCII code of `ch` is printed first in decimal, right-justified in a 3-character field without zero-fill, and then again in hex, prefixed by "0x", followed by a right-justified 0-filled hex number in a 2-character field. If `ch` is printable as defined by `isprint()`[1], then it should also be printed as a character, enclosed in single quotes as shown.

For example, if file `data.in` contains the text:

```
Score was 35to21.
```

the output should be:

```
------------------------------------------------------------
        Ima Goetting Closeau
        CPSC 427/527
        Tue Oct  4 2016 11:18:06
------------------------------------------------------------
Skipping char:  83 0x53 'S'
Skipping char:  99 0x63 'c'
Skipping char: 111 0x6f 'o'
Skipping char: 114 0x72 'r'
Skipping char: 101 0x65 'e'
Skipping char: 119 0x77 'w'
Skipping char:  97 0x61 'a'
Skipping char: 115 0x73 's'
35
Skipping char: 116 0x74 't'
Skipping char: 111 0x6f 'o'
21
Skipping char:  46 0x2e '.'
Loop exit
------------------------------------------------------------
Normal termination.
```

Be sure you understand why there is no "Skipping" line for the spaces following "Score" and "was". What happened to those characters?

3 Programming Notes

This program is very short and may be put entirely in the `run()` function in `main.cpp`.

You *must* read `x` using the stream extract operator `>>`. You may not use `stringstream` or `getline()` or other methods to read the line as a string or to read individual digits that comprise a decimal number. You must let the stream do your decimal to binary conversion. Do not call `atoi()` or `strtol()` or any other means of manually converting a string to an `int`.

To obtain the ASCII code of a character stored in a `char` variable `ch`, cast `ch` to an `int`. Similarly, to print a character whose ASCII code is stored in an `int` variable `x`, cast `x` to a `char` before printing.

[1] See http://www.cplusplus.com/reference/cctype/isprint/.

Grading Rubric

Your assignment will be graded according to the scale given in Figure 1 (see below).

#	Pts.	Item
1.	1	All relevant standards from PS1 are followed regarding submission, identification of authorship on all files, and so forth.
2.	1	A well-formed `Makefile` or `makefile` is submitted that specifies compiler options `-O1 -g -Wall -std=c++17`.
3.	1	Running `make` successfully compiles and links the project and results in an executable file `readint`.
4.	1	Your program gives a usage comment and terminates if the wrong number of command line arguments are given. It gives a descriptive error comment if the specified input file does not open.
5.	4	All instructions given in sections 2 and 3 are carefully followed.
6.	4	Your program correctly extracts all of the integers in the file.
7.	4	Your program prints a correct "Skipping…" message following each failed attempt to read an integer.
8.	2	The "Skipping…" message exactly follows the examples and instructions, including spacing and when to print leading 0's and when not to.
9.	2	Your program correctly handles end-of-file, regardless of whether the EOF is immediately preceded by whitespace, a digit, or another character.
	20	Total points.

Figure 1: Grading rubric.