Command Line Arguments - Advanced

Topics / Objectives

1. Detecting the presence of command-line options (what command line options, if any, did the user provide on your program's command line?)
2. Command line validation (did the user enter a valid command line?)
3. C arrays
4. C++ std::vector<T> template class
5. Using C++'s exception handling feature to manage runtime errors (deriving your own custom exception classes from C++'s standard exception classes)
6. C++ std::string class (working with C++ strings)
7. C++ file I/O streams (reading from and writing to files)
8. C++ string-based I/O streams (reading from and writing to C++ strings)
9. Function pointers and callback functions

Start by creating a folder named ece3220-lab06-$USER in your $HOME directory, where $USER is your pawprint:

```
$ mkdir ~/ece3220-lab06-$USER
$ cd ~/ece3220-lab06-$USER
```

For this lab you will write a program that accesses numeric data stored within text files whose contents are similar to those shown for Listing 1. The first line in each data file has two integer values separated by one or more space characters, or tab characters, or both. The first integer indicates the data sample size—e.g., the value '5' indicates this file contains five data samples. The second value '15' represents the maximum allowed value of any data sample within this sample set. This maximum value might or might not appear in the data. Any data sample whose value exceeds the maximum value must be considered invalid data by your program. Each remaining line in the file (2, 11, 0, 8, and 13) corresponds to one sampled data value.

```
5  15
2
11
0
8
13
```

Listing 1: Information elements in a sample data file.

On Canvas, the assignment page for this lab provides a link to a gzipped tarball file named **`validate.tar.gz`**. This tarball file provides the test data you must use to validate the program you create for this lab. Download a copy of this tarball, unpack its contents within your ece3220-lab06-$USER directory, and then note that you now have a subdirectory named **validate** whose structure is as follows:

```
$ cd ~/ece3220-lab06-$USER
$ tar xf <PATH_TO>/validate.tar.gz
$ tree -d
.
`-- validate/
    |-- raw-data-invalid/
    |   |-- raw_data_??.txt
    |   `-- README
    `-- raw-data-valid/
        `-- raw_data_??.txt
```

The subdirectory **validate/raw-data-invalid/** contains files whose contents are invalid in some way. Use these data files to verify the error detection and error reporting code paths in your lab6 program.

The subdirectory validate/raw-data-valid/ contains data files that are valid. Use these data files to verify that your lab6 program correctly processes valid data files.

Your program must accept user-supplied command line arguments as described below. Recall that the operating system passes command line information into your program via function main()'s 'argc' (argument count) and 'argv' (argument vector) arguments:

```
int  main(int argc, char *argv[])
```

The command line arguments your program must recognize and use are shown below. Explanations on the different options/operations will be given further down.

```
-n FILENUM        // File number (value needed)
-a OFFSET         // additive offset value (value needed)
-s SCALE_FACTOR   // multiplicative scale factor (value needed)
-h                // Prints a help message and exits.
                  // This message shall display a synopsis of
                  // valid command lines, and shall provide a
                  // description of all options.
-d DATA_DIR       // The path to the folder that contains the
                  // data files.
```

Assume you name your program ‘lab6’. The SYNOPSYS section below indicates the two possible command lines for the program you will create for this lab. These are the only valid command lines for your program.

SYNOPSYS

lab6 -n *FILENUM* [**-a** *OFFSET*] [**-s** *SCALE_FACTOR*]

lab6 -h

OPTIONS

-a *OFFSET*

The specified *OFFSET* value is added to each sample data value read from input file *raw_data_NN.txt*. The transformed data values are written into a file named *offset_data_NN.txt*, where *NN* is a two digit, zero-padded integer value whose value is specified by the **-n** option. For example, for **-n 3**, NN’s value is 03, and program **lab6** writes the transformed data into a file named *offset_data_03.txt*.

-h

Displays a useful message that describes how to use this program.

-n *FILENUM*

FILENUM is an integer value between 1 and 99 inclusive. Program **lab6** reads data from a text file named *raw_data_NN.txt* where *NN* is a two digit, zero-padded integer value whose value is specified by *FILENUM*. For example, if *FILENUM*’s value is 3, then NN’s value is 03, and program **lab6** reads from file *raw_data_03.txt*.

-s *SCALE_FACTOR*

Each sample data value read from input file *raw_data_NN.txt* is multiplied by the specified SCALE_FACTOR value. The transformed data values are written into a file named scaled*_data_NN.txt*, where *NN* is a two digit, zero-padded integer value whose value is specified by the **-n** option. For example, for **-n 3**, NN’s value is 03, and program **lab6** writes the transformed data into a file named *scaled_data_03.txt*.

-d *DATA_DIR*

If the raw data files *raw_data_NN.txt do not reside* in the current working directory, use this option to specify the path to the directory where the *files reside.* If this program generates new files that contain transformed data, those new files will be stored in the DATA_DIR *directory*.

The following examples illustrate how to interpret the options above.

```
$ ./lab6 -n 3 -a 2.5 -d ./validate/raw-data-valid
```

Your program reads the file "raw_data_03.txt" and it adds an offset value of 2.5 to each sampled signal value read from that file. The transformed signal data is stored in a new output file whose file name is 'offset_data_03.txt".

```
$ ./lab6 -s 1.7 -n 10 -d ./validate/raw-data-valid
```

Your program reads the file "raw_data_10.txt" and it scales (multiplies) each sampled signal value by 1.7. The transformed signal data is stored in a new output file whose file name is 'scaled_data_03.txt".

```
$ ./lab6 -n 2 -a 4.3 -s 5  -d ./validate/raw-data-valid
```

Your program reads the file "raw_data_02.txt" and first it scales (multiplies) each sampled signal value by 5, and it writes each scaled value into an output file whose file name is 'scaled_data_02.txt". Second, it adds the offset value 4.3 to each sampled signal value it reads, and it writes the result value into an output file whose file name is 'offset_data_02.txt'.

```
$ ./lab6 -h [...]
```

If the **-h** option is supplied, your program must ignore any/all other command line options, and your must program display a message that explains how your program should be invoked, to include the complete set of command line options. The program then exits with an exit status of zero.

Save any transformed signal data into a new file. Data values that are transformed by adding an offset value shall be saved into a file named "offset_data_NN.txt" where **NN** is the same two-digit zero-padded number as the raw file—e.g., raw_data_03.txt → offset_data_03.txt. Likewise, data values that are transformed by multiplying by a scale factor value shall be saved into a file named "scaled_data_NN.txt"--e.g., raw_data_25.txt → scaled_data_25.txt.

These output files shall have the following format. The first row will two integer values and separated by a single space. The first value on the first row must be a positive integer that identifies the number of data values in the file (the data set size). The second value on the first row shall be the OFFSET value for offset_data_NN.txt files, or the SCALE_FACTOR value for scaled_data_NN.txt files. The remaining rows, in both the offset and scaled output files, will have the transformed values with a single value per line. All floating point “double” values shall be written with 4 decimal points.

Consider the example shown in Listing 2. The program name is ‘**lab6**’. The user-selected data file is **raw_data_03.txt** as specified via the command line option ‘**-n 3**’. The values shown under the column heading **offset_data_03.txt** are the transformed values written into a file of the same name, with an offset value of -1.2 as specified via the command line option ‘**-a -1.2**’. Likewise, the values shown under the column heading **scaled_data_03.txt** are the transformed values written into a file of the same name, with a scale factor value of 2.75 as specified via the command line option ‘**-s 2.75**’.

```
$ ./lab6 -a -1.2  -n 3  -s 2.75 -d ./validate/raw-data-valid

raw_data_03.txt     offset_data_03.txt       scaled_data_03.txt
/* -n 3 */          /* -a -1.2 */            /* -s 2.75 */
5  15               5  -1.2                  5  2.75
2                   0.8000                   5.5000
11                  9.8000                   30.2500
0                   -1.2000                  0.0000
8.5                 7.3000                   23.3750
13.98               12.7800                  38.4450
```

Listing 2: Example that shows the desired output values for an offset value of -1.2 and a scale factor value of +2.75.

:: IMPORTANT ::

The program’s command line options may be provided in any order. Furthermore, some options have a required parameter—e.g., -n FILENUM—while others do not—e.g., -h .

Your program must detect and manage invalid input. Consider the following examples.

```
# ERROR: Required option ‘-n FILENUM’ is missing
$ ./lab6
```

```
# ERROR: Option '-n' is missing the required 'FILENUM'
#        parameter
$ ./lab6 -s 17 -n
```

In each of these cases your program does not have enough information to continue and must respond as follows:

1. It must print to **stderr** an error message that indicates the detected problem, and
2. It must print to **stderr** the following message,

   ```
     Try './lab6 -h' for more information.
   ```

   and
3. the program must exit with a non-zero exit status code.

In a nutshell your program will proceed as follows:

1. You must use C++'s exception handling feature to manage all runtime errors.
2. Detect which command line options, if any, are present on your program's command line.
3. Determine whether the user entered a valid command line, as shown under the SYNOPSYS section on page 3.
4. If the command line option '-h' is present, display a help message that instructs the user how to properly use the program (e.g., the help text shown on page 3), and then exit the program with an exit code of zero (success).
5. Else
   1. Read and validate the data values from input raw_data_NN.txt input file, where NN is specified via the command line option **-n FILENUM** (i.e., FILENUM → NN)
   2. If the command line option **-a OFFSET** is present, transform the raw data by adding OFFSET to each raw data value and store the transformed data values into a file named offset_data_NN.txt, and
   3. If the command line option **-s SCALE_FACTOR** is present, transform the raw data by multiplying each raw data value by SCALE_FACTOR and store the transformed data values into a file named offset_data_NN.txt.

Deliverables

1. Demonstration: The TA will ask you to run some examples, to demonstrate that your programs are working properly.
2. Create a gzipped tarball of your folder ~/ece3220-lab06-$USER/, and use the SUBMIT ASSIGNMENT button provided with this assignment on Canvas to upload a copy of your gzipped tarball file.

Grading:

Demo:	50	
Code:	50	**(Well organized, properly commented and indented)**