21.1 Lab 1: Data visualization

In this assignment you are going to handle some basic input operations including validation and manipulation, and then some output operations to take some data and format it in a way that's presentable (i.e. readable to human eyes).

Functions that you will need to use:

`getline(istream&, string&)`
This function allows you to get input for strings, including spaces. It reads characters up to a newline character (for user input, this would be when the "enter" key is pressed). The first parameter is typically `cin`, with the second parameter a string you want to read, like this:

```
string input;
getline(cin, input);
```

`setw(int)`
This looks like a function, but is really a stream manipulator. It allows you to specify how many characters the next output should be. This is useful when trying to line things up different outputs. (For more information look at section 12.1 Output formatting)

`int stoi(string&)`
This function takes a string, converts it to an integer and returns that integer. For example, the string "-24" returns an integer with the value of -24. If it is unable to convert (i.e. the string contains "Batman"), an exception of type `invalid_argument` is thrown, which you will need to catch if you want your program to continue.

`try/catch`
Not functions, but keywords, these are used to handle exceptions. Sometimes operations can fail to generate the correct results, while other times they may fail to generate ANY result. This could be due to incorrect input, going out of bounds of an array, and a variety of other cases. For more information, look at section 15.1 Exception basics.

Input Processing

Handling input will require the following steps:

- Get input from the user in the form of a string (see the getline() function, above)
- Check to see if that string has ONE comma in it
- Split the string based on that comma into two parts
- Convert the second part into an integer

Input Validation

For strings, validation is typically handled by checking to see if a string is equal to something, or perhaps whether it contains a certain character, or number of characters, etc.

For numeric values, though, while we typically validate whether its in a certain range or not, there is the possibility of something not being a number to begin with (for example, the user enters "dinosaur" when prompted to enter a number).

Assignment Details

(1) Prompt the user for a title for data. Output the title. (1 pt)

Ex:

```
Enter a title for the data:
Number of Novels Authored
You entered: Number of Novels Authored
```

(2) Prompt the user for the headers of two columns of a table. Output the column headers. (1 pt)

Ex:

```
Enter the column 1 header:
Author name
You entered: Author name

Enter the column 2 header:
Number of novels
You entered: Number of novels
```

(3) Prompt the user for data points. Data points must be in this format: *string, int*. Store the information before the comma into a string variable and the information after the comma into an integer. The user will enter **`-1`** when they have finished entering data points. Output the data points. Store the string components of the data points in a **vector of strings**. Store the integer components of the data points in a **vector of integers**. (4 pts)

Ex:

```
Enter a data point (-1 to stop input):
Jane Austen, 6
Data string: Jane Austen
Data integer: 6
```

(4) Perform error checking for the data point entries. If any of the following errors occurs, output the appropriate error message and prompt again for a valid data point. For the checking whether the entry after the comma is an integer or not, you will have to use `try/catch` blocks with the `stoi` function

- If entry has no comma
  - Output: `Error: No comma in string.` (1 pt)
- If entry has more than one comma
  - Output: `Error: Too many commas in input.` (1 pt)
- If entry after the comma is not an integer
  - Output: `Error: Comma not followed by an integer.` (2 pts)

Ex:

```
Enter a data point (-1 to stop input):
Ernest Hemingway 9
Error: No comma in string.

Enter a data point (-1 to stop input):
Ernest, Hemingway, 9
Error: Too many commas in input.

Enter a data point (-1 to stop input):
Ernest Hemingway, nine
Error: Comma not followed by an integer.

Enter a data point (-1 to stop input):
Ernest Hemingway, 9
Data string: Ernest Hemingway
Data integer: 9
```

(5) Output the information in a formatted table. The title is right justified with a setw() value of 33. Column 1 has a setw() value of 20. Column 2 has a setw() value of 23. (3 pts)

Ex:

```
        Number of Novels Authored
Author name         |       Number of novels
--------------------------------------------
Jane Austen         |                      6
Charles Dickens     |                     20
Ernest Hemingway    |                      9
Jack Kerouac        |                     22
F. Scott Fitzgerald |                      8
Mary Shelley        |                      7
Charlotte Bronte    |                      5
Mark Twain          |                     11
Agatha Christie     |                     73
Ian Flemming        |                     14
J.K. Rowling        |                     14
Stephen King        |                     54
Oscar Wilde         |                      1
```

(6) Output the information as a formatted histogram. Each name is right justified with a setw() value of 20. (4 pts)

Ex:

```
         Jane Austen ******
     Charles Dickens ********************
    Ernest Hemingway *********
        Jack Kerouac **********************
 F. Scott Fitzgerald ********
        Mary Shelley *******
    Charlotte Bronte *****
          Mark Twain ***********
     Agatha Christie 
******************************************************************


        Ian Flemming **************
        J.K. Rowling **************
        Stephen King 
******************************************************
         Oscar Wilde *
```

Blind Tests (remaining points)

These are variations on the previous tests, with hidden input. Your code should work exactly the same in a variety of scenarios. For example, if you were to remove Mark Twain from the previous example, everything else should work just fine and the test would pass without issue.

327664.1761198.qx3zqy7

LAB ACTIVITY | 21.1.1: Lab 1: Data visualization 30 / 30 ✓

main.cpp Load default template...

```
#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
using namespace std;

int main()
{
   //**********VARIABLE DECLARATIONS**********
   string title;
   string column1;
   string column2;
   string dataPoint;
   int firstComma = 0;
   vector <string> wordInputs;
   vector <int> numberInputs;
   string strData;
   int intData = 0;
   int commaCount;
```

Develop mode **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.cpp** (Your program)

Program output displayed here

Signature of your work What is this?

```
5/22.. S------------------------------------------------
-----------------------|19|12|18|8|9|4|0|0|4|4|4|4|0|2
|12|4----|10|10|5|5|12|12|12|12----|2------|2-----|12|12
|26|26|2|26|26 U-----|30|30- ..5/23
```

Trouble with lab?