Lab 1

Lab 1

As you read a text file, you go line by line until you reach the end of the file. What happens if you want to go back to a specific line of text? A common practice that comes with reading text from a file is to store that information into something like a vector. This way you can easily reference any data obtained from the file.

Before reading the file, create the `path` variable with the file path, and instantiate the string vector `text`.

```
string path = "student/labs/fileslab1.txt";
vector<string> text;
```

Use `try`, `throw` and `catch` blocks to handle input/output exceptions. In the `try` portion, create an `ifstream` object to read through the file and store its content into a string variable. While reading, add each line to the vector `text`. Print any errors in the `catch` portion. You can optionally print a message that the file has successfully been read, but that is not required.

```
string path = "student/labs/fileslab1.txt";
vector<string> text;

try {
  ifstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  while (getline(file, read)) {
    text.push_back(read);
  }
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

The content of the text file now resides in the vector variable `text`. However, the code above only adds text from the file into the vector. To print what was stored in the vector, use a loop to iterate the vector's elements first followed by the `cout` command.

```
string path = "student/labs/fileslab1.txt";
vector<string> text;

try {
  ifstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  while (getline(file, read)) {
    text.push_back(read);
  }
  for (int i = 0; i < text.size(); i++) {
    cout << text.at(i) << endl;
  }
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

You should see a passage from Bram Stoker's *Dracula*. You'll notice, however, that the output is just a collection of text grouped together. In fact, if you were to print the first element in the vector, you will get the same result. The entire file was read and stored as the first element in the vector. This occurs because the default delimiter is a newline and there is only 1 occurrence of a newline at the end of the file.

Let's change the delimiter into a period `.` so that the text will be tokenized into sentences. Each token will represent one sentence from the passage.

```
string path = "student/labs/fileslab1.txt";
vector<string> text;

try {
  ifstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  while (getline(file, read, '.')) { //set delimiter as a period
    text.push_back(read);
  }
  for (int i = 0; i < text.size(); i++) {
    cout << text.at(i) << endl;
  }
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

The passage is tokenized using a period `.` as a delimiter. When the delimiter is specified, the system extracts the period from the text, this is why you do not see the periods in the output. To put the periods back into the vector, simply include `+ '.'` in the `push_back()` statement.

```
string path = "student/labs/fileslab1.txt";
vector<string> text;

try {
  ifstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  while (getline(file, read, '.')) {
    text.push_back(read + '.'); //add period to end
  }
  for (int i = 0; i < text.size(); i++) {
    cout << text.at(i) << endl;
  }
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

Your vector now includes 4 elements, each representing a sentence from the passage extracted from the file `fileslab1.txt`. To bring your focus to a particular sentence, you can use the `at()` function and specify the position of the sentence you are interested in.

```
string path = "student/labs/fileslab1.txt";
vector<string> text;

try {
  ifstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  while (getline(file, read, '.')) {
    text.push_back(read + '.');
  }
  cout << text.at(1); //print the second element/sentence
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

To erase the first leading white space, you can use `text.at(1).erase(0, 1)`. This will take the system to the first position, index `0`, and erase just `1` character in that string.

```
string path = "student/labs/fileslab1.txt";
vector<string> text;

try {
  ifstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  while (getline(file, read, '.')) {
    text.push_back(read + '.');
  }
  cout << text.at(1).erase(0, 1) << endl; //erase the first
        string char
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

Lab 2

Lab 2

This lab uses a comma delimited CSV file `fileslab2.csv`, which contains integers. There are three columns and four rows. The program below will print the sum for each row in the CSV. This is what the file currently looks like:

```
1,4,5
18,34,99
0,12,51
37,29,61
```

We'll start with directing the `path` to the file, creating a vector `nums` to store the data for later, creating an `ifstream` and string to read and hold the content of the file temporarily, and using `try`, `throw` and `catch` blocks to handle any issues when opening the file.

```
string path = "student/labs/fileslab2.csv";
vector<string> nums;

try {
  ifstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

After, use the `getline()` function to iterate through the file and store its content as tokens in the vector. Note that you will need to go through the file twice and include a `stringstream` object to further help tokenize the data. Try running the code below to see what's currently stored.

```
string path = "student/labs/fileslab2.csv";
vector<string> nums;

try {
  ifstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  while (getline(file, read)) {
    stringstream ss(read);
    while (getline(ss, read, ',')) {
      nums.push_back(read);
    }
  }
  for (int i = 0; i < nums.size(); i++) {
    cout << nums.at(i) << endl;
  }
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

You should see a list of numbers after the code runs. Though what you see are numbers, they are currently strings which means we cannot do calculations on them directly. We must convert them into integers first using `stoi()`. Additionally, the data is stored in a vector which is one dimensional but we want to calculate totals for multiple rows. To achieve, we'll use a double nested `for` loop to iterate through the vector in chunks of **three** elements which will allow us to calculate the totals of each row.

```
string path = "student/labs/fileslab2.csv";
vector<string> nums;

try {
  ifstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  while (getline(file, read)) {
    stringstream ss(read);
    while (getline(ss, read, ',')) {
      nums.push_back(read);
    }
  }
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}

for (int i = 0; i < nums.size(); i+=3) {
  int total = 0;
  for (int j = 0; j < 3; j++) {
    total += stoi(nums.at(i + j));
  }
  cout << "Total: " << total << endl;
}
```

Your program should print the following output:

```
Total: 10
Total: 151
Total: 63
Total: 127
```

Lab 3

Lab 3

The goal of this lab is to rely on user input for data. We are going to continuously ask the user to enter the name of a superhero followed the name of their power. If the user enters lowercase `q`, the system will stop collecting data and write all of the data collected to the CSV file `superheroes.csv`.

First let's create our `string` path, `ofstream` object, `read` string, and exception blocks like usual.

```
string path = "student/labs/superheroes.csv";

try {
  ofstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }
  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

Next, we need to create variables for our user input. Additionally, we also need to continuously ask the user for output until they enter `q`. After `q` is detected, the information entered will be written to our CSV file.

```
string path = "student/labs/superheroes.csv";
string name;
string power;

try {
  ofstream file;
  string read;
  file.open(path);
  if (!file) {
    throw runtime_error("File failed to open.");
  }

  while (true) {
    cout << "Please enter a superhero name (or enter q to quit):
        ";
    cin >> name;
    if (name == "q") {
      break;
    }
    cout << "Please enter a superhero power (or enter q to
        quit): ";
    cin >> power;
    if (power == "q") {
      break;
    }
    file << name << ',' << power;
  }

  file.close();
}

catch (exception& e) {
  cerr << e.what() << endl;
}
```

Click the `TRY IT` button to enter information into the terminal. Enter `q` to stop the data collection and write to to the CSV file. Click on the CSV file link below to see everything that was entered.

Open superheroes.csv

Lab Challenge

Lab Challenge

Write a program that reads a text file . This file is stored in the variable `path`.

DO NOT alter this line of code in the program!

```
///////// DO NOT EDIT! /////////
  string path = argv[1];        //
////////////////////////////////
```

The file contains several instances of the word `Burma`. Replace each instance of `Burma` with `Myanmar`, and print the results of this transformation. The final output of your program should be:

```
Myanmar is a country in Southeast Asia.
The capital of Myanmar is Naypyidaw.
Its population is about 54 million people.
Myanmar is a former British colony.
```

▼ **Hint**

You can use the `FindAndReplace()` function to replace all instances of `Burma` with `Myanmar`. Note that you will need to store the content of the file into a string in order to use this function.