Url Parser Project

URL's are used to represent resource locations on the internet. Understanding URL's is one of the fundamental requirements of any program that interacts with online resources, including web browsers and mobile applications. For this project, we will write some code in C++ that can help us disect a url string into it's different parts. We will be working with a slightly simplified version of valid *absolute* URL's.

Project Background

Parts of a URL

There are many types of URL's. We will be working with a subset most commonly used in online applications. Here are a couple of example URL's:

- `http://example.com/path/to/file.txt?test=true&value=three#fragment-string`
- `https://www.example.com?test=true`

In general, a URL can be broken down into the following pieces:

`<scheme>://<location>/<path>?<query>#<fragment>`

The `<scheme>` consists of anything preceeding the first `:` character.

The `<location>` consists of everything between the first two `/` characters, and the third `/` character. The third `/` may not be present, i.e. `https://test.com`.

The `<path>` is everything between (and including) the third `/` character and before the `?` character.

The `<query>` is anything following the `?` character and preceeding the `#` character.

The `<fragment>` is anything following the `#` character.

Disecting a URL

The URL parser should be able to take a url provided as a text string, and extract each of the component parts from it. For instance, given the url: `https://example.com/file.txt#para1`, the parser should identify:

- `<scheme> = https`
- `<location> = example.com`
- `<path> = file.txt`
- `<query> =`
- `<fragment> = para1`

Note that there is no `<query>` in this particular example.

Project Requirements

You will write your own `C++` project from scratch. You may use the following snippet to get started:

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

int main()
{
    string url;
    cout << "Please enter a url: ";
    cin >> url;

    cout << "The url is: " << url << endl;
    cout << "<scheme> = " << endl;
    cout << "<netloc> = " << endl;
    cout << "<path> = " << endl;
    cout << "<query> = " << endl;
    cout << "<fragment> = " << endl;
}
```

Please make sure to cover the following requirements:

1. An absolute url should have a `scheme` and a `netloc`. If either or both of these are missing, please print out an error message such as *the url was invalid*.
2. Disect and print out each of the parts of a url including the `scheme`, `netloc`, `path`, `query`, and the `fragment`. If one of these parts is missing, printing out an empty line is ok, i.e. `<path> =`
3. Your program should not crash on any input including empty (no input at all) and strings of completely random characters, i.e. `ajfdq832jf9e*#(*#*())`. In this latter example, the correct output would be the error message from requirement #1 because there is not `scheme` or `netloc`.
4. Organize the code well using functions, do *not* put the entire program inside of `main`.
5. Do not share code with classmates. It is perfectly fine to discuss the project and strategies with each other, as well as look up and integrate code snippets from online resources.

Implementation Steps

You are free to implement this program however you wish, but I suggest the following order:

1. Parse the fragment, then remove it from the url string. Also remove the `#` character.
2. Parse the query, then remove it from the url string. Also remove the `?` character.
3. Parse the scheme, then remove it from the url string. Remove the `://` characters. Be careful to only remove characters that are actually present in the string.
4. Parse the netloc, then remove it from the url string.
5. What remains is the path.
6. Make sure the input is a valid url and report an error otherwise.
7. Make sure random strings don't crash your program.

Grading Breakdown (100 points)

- **(15)** The program correctly identifies the `scheme`
- **(15)** The program correctly identifies the `netloc`
- **(15)** The program correctly identifies the `path`

- **(15)** The program correctly identifies the `query`
- **(15)** The program correctly identifies the `fragment`
- **(5)** The program correctly identifies an invalid url
- **(10)** The program doesn't crash for any input
- **(10)** The code is well organized and not all inside of `main`

Submitting the project

I would like you to submit the entire project source. There are a couple ways to do this. On Canvas, you can upload a .zip archive of the project directory. Alternatively, you can submit a link to a location online where you have uploaded the code (like Github for example).

Useful Resources

- LearnCpp C++ Strings
- cppreference std::string