Author: Zesheng Xu

NetID: ZXX190011

Date: 9/20/2021

Introduction:

This program is an URL parser that breaks down input URLs into 4 components: protocol, domain, path, and parameters. It is able to automatically locate the said component by their respective distinct features, Which also gives it robustness against edge cases to a certain extent i.e missing components or even missing important segmentation symbols.

Code breakdown:

Global variables:

I created 4 bool variables that are global. They are dedicated to reflecting if any components of the URL are erroneous, which are reflected through their naming i.e protocol\_error.

Main :

This is the main function that calls sub-functions that parse different parts of the URL. It is also responsible for all of the user interactions.

First, a few variables are created to contains the message from the functions that are responsible for parsing different components of the URL, they will be assigned value to the returns of the respective function.

Then, in an, if statement, if any bool variables that are meant to reflect any erroneous part are true, we will display the corresponding message that is generated from the parsing functions. If no error is given, we will proceed to display all messages containing variables that contain the breakdown of the URL.

Sub-functions:

protocol \_parsing:

2 string variables are created, they both attempts to locate the protocol through 2 characters the “//” and “:”. If they do not return the same result, we will terminate the function and return an error message.

If they return the correct answer, indicating that the protocol is formatted correctly, we will proceed to check the protocol itself.

If the protocol is within the one of the accepted protocols, we will return the broken down protocol, if not we will return an error statement stating it is an invalid protocol

Domain\_parsing :

At the beginning of the function,we first remove the protocol from the URL by locating the “//” symbol and doing a substring. Then we locate the end of domain by either an “:” or “/” depends on if a “:” exists, which is part of the port component.

Then with the domain extracted from the URL, through a for loop, we count how many “.” there are, ensuring it is properly formatted.

At last, we simply compare its ending to the acceptable domains, if it is within one of the accepted ones, we will return it, if not we will return an error message stating it is invalid.

Port\_parsing:

At the beginning, we locate the last “:” sign, which indicates the location of the port. If its return location value is < 8, that means it is the “:” from protocol, thus a port do not exist and we will return an error statement.

If the port was able to be located through such a method, we proceed to check if all chars in the string are integers, if not, we return an error statement.

Then we proceed to check if the port is a valid port by turning the String into int through stoi() method. If the port value is greater than one and smaller than 65535, we will return its value, else we will return an error statement stating the accepted port range.

Path\_parsing:

At the beginning of the function, we first remove the protocol from the URL by locating the “//” symbol and doing a substring. Then we locate the end of the path by either an “?” or simply til the end of the URL, depends on if a “?” exists, which is part of the parameter component.

Then we simply check if it starts with an “/’ symbol and ends with either html or htm. If it does, we return it, else we return an error statement stating it is invalid.

Parameter\_parsing:

We first make sure a “?” is present in the URL, if not, we return a statement of “parameter not given”. If it exists, we simply do a substring of the URL from the “?” to its end, and return the value.