In this laboratory exercise you will build a simple webcrawler for searching all the pages of a specified Web site. This is a simple version of spidering, which is what search engines do as they build their Web dictionaries.

Screen Scraper - A screen scraper retrieves all the text from a Web page and loads it into a byte array. In our first program demo we will create a C# console application named WebDataMiner with a main method as shown below:

```csharp
static void Main(string[] args)
{
    string strUrl;
    Console.Write("Enter URL... ");
    strUrl = Console.ReadLine();
    strUrl = "http://" + strUrl;
    WebClient webClient = new WebClient();
    byte[] reqHTML;
    reqHTML = webClient.DownloadData(strUrl);
    UTF8Encoding objUTF8 = new UTF8Encoding();
    string html = objUTF8.GetString(reqHTML);
    Console.WriteLine(html);
    Console.ReadKey();
}
```

Before compiling this project you will need to add the System.Net namespace to the list of using statements.

```csharp
using System.Net;
```

The System.Net namespace provides a simple programming interface for many of the protocols used on networks today. The WebRequest andWebResponse classes form the basis of what are called pluggable protocols, an implementation of network services that enables you to develop applications that use Internet resources without worrying about the specific details of the individual protocols.

The string strUrl holds the URL for the Web site. When the user enters the target URL the http:// header should be omitted since it is automatically added to the beginning of the URL by,

```csharp
strUrl = "http://" + strUrl;
```

The byte array reqHTML receives the text of the target Web page using the DownloadData method of WebClient class.

```csharp
reqHTML = webClient.DownloadData(strUrl);
```
The `objUTF8` object is an instance of the `[UTF8Encoding](https://docs.microsoft.com/en-us/dotnet/api/system.text.UTF8Encoding)` class.

```csharp
    UTF8Encoding objUTF8 = new UTF8Encoding();
```

This class is based on the **UTF-8** *(8-bit UCS/Unicode Transformation Format)* a variable-length character encoding for Unicode. It is able to represent any character in the Unicode standard, yet is backwards compatible with **ASCII**. For these reasons, it is becoming the preferred encoding for e-mail, web pages, and other places where characters are stored or streamed. **UTF-8** encodes each character (code point) in 1 to 4 octets (8-bit bytes), with the single octet encoding used only for the **128 US-ASCII** characters.

The unicode held in `objUTF8` can be converted into a text string, `html` using the `GetString()` method of the `[UTF8Encoding](https://docs.microsoft.com/en-us/dotnet/api/system.text.UTF8Encoding) `class,

```csharp
    string html = objUTF8.GetString(reqHTML);
```

This string can be displayed passed to a browser or scanned by a string-matching filter to find objects or sub-strings of interest. In our application we will be searching for and extracting all the static links to other Web pages.

Run the program using the url,

```csharp
    csclab.murraystate.edu/bob.pilgrim
```

Verify that the HTML source for this Web page is displayed in the command prompt console.

**A Link Filter** - Now we are ready to build a link filter. In HTML a link is preceded by the character sequence, a `href` and ends with a second double quote, `. The next modification to WebDataMiner will be a link filter. We are searching for links to pages associated with this Web site, so we will use the string `"href="` as our search key.

We will add a class-level (i.e. global) list variable to the project to hold the links we find,

```csharp
    public static List<string> hrefList = new List<string>();
```

The use of the C# list type has some advantages over a standard array of strings. In particular we have access to the methods and properties of the `List<>` permitting us to Add members, count the members and access the members using a `foreach` statement or by index. Examples:

```csharp
    hrefList.Add(a_str);

    foreach (string str in hrefList)
        Console.WriteLine(str);

    Console.WriteLine(" num links found = 
```
For the fun of it (actually for consistency) we will modify the main program so that the initial URL is the first link in the list of links.

```csharp
static void Main(string[] args)
{
    string str;
    Console.Write("Enter URL... ");
    str = Console.ReadLine();
    str = "http://" + str;
    hrefList.Add(str);
    WebClient webClient = new WebClient();
    byte[] reqHTML;
    reqHTML = webClient.DownloadData(hrefList[0]);
    UTF8Encoding objUTF8 = new UTF8Encoding();
    string html = objUTF8.GetString(reqHTML);
    Console.WriteLine(html);
    Console.ReadKey();
}
```

Now we will add a method called `hrefFilter()` to search for the links in the HTML code stored in the string variable `html`. As the string is passed into `hrefFilter()` it becomes `str`.

```csharp
public static void hrefFilter(string str)
{
    int num;
    for (int i = 0; i < str.Length - 7; i++)
    {
        string a_str = "";
        if (str.Substring(i, 5) == "href=")
        {
            num = i + 6;
            while (str[num] != '"')
            {
                a_str = a_str + str[num];
                num += 1;
            }
            hrefList.Add(a_str);
        }
    }
}
```

Inside the `for` loop the counter `i` moves character-by-character through `str` looking for the substring "href=". When there is a match, the `while` loop accumulates the characters between the double quotes into a string named `a_str`.

The integer `num` keeps up with the sequence of characters in this scan. Upon exit from the `while` loop `a_str` is added to the list `hrefList`. Note that the initial value of `num` is set to the current value of `i` plus 6, which gets us past the initial double quote (assuming that there are not extraneous spaces in the HTML string).

From the previous description is should be clear that our simple filter is not very robust. This is not an issue for the demonstration purposes of this laboratory exercise, however any serious attempt at building a Web data mining project would need to consider as many variations in HTML text formatting as possible.
We will include the method `show_hrefList()` in order to view the list of links found.

```csharp
public static void show_hrefList()
{
    Console.WriteLine(" num links = {0}", hrefList.Count());
    foreach(string str in hrefList)
        Console.WriteLine(str);
}
```

Finally we replace the `Console.WriteLine(html)` in the main program with calls to the two new methods,

```
hrefFilter(html);
show_hrefList();
```

Since the list `hrefList` is declared as the class-level, it is within the scope of `main()` as well as the methods `hrefFilter()` and `show_hrefList()`.

**Completing WebDataMiner** - To complete this project we would

1. scan the list of links,
2. find those links that refer to Web pages at the same site,
3. capture each of these pages,
4. append links found in these pages to `hrefList`

Local links are usually defined relative to the current page so they will not include the full URL address. Also, depending on the application, we may want to skip links to other files (i.e. those that do not end with ".htm" or ".html". The details of recognizing and avoiding links to other sites, reconstructing URL's for local pages and the handling of non-HTML links is beyond the scope and intent of this exercise.

**Questions**

1. How would you recognize links to other sites? ________________________________
   __________________________________________________________________________

2. How would you separate links to HTML pages from links to files or other objects?
   __________________________________________________________________________
   __________________________________________________________________________

3. How would you reconstruct a link to another page on the Web site being scanned when the links provided do not include a complete URL? ________________________________
   __________________________________________________________________________

4. Briefly discuss the complications for Web data mining created by JavaScript, PHP and other dynamic Web based languages. ________________________________
   __________________________________________________________________________
   __________________________________________________________________________