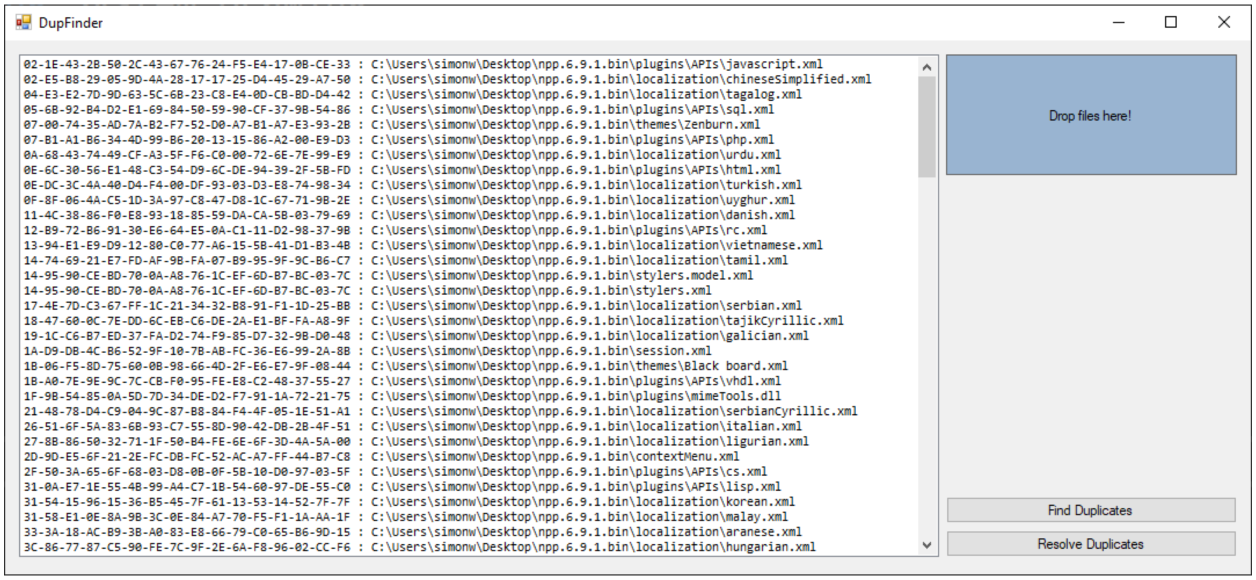
CMPE1666 – Lab #4 – DupFinder

In this lab you will determine if selected files have exactly the same contents. You will use a variety of new framework functions and types that you may not have seen before. Because of this, the lab will take you through these new types, and suggest techniques to do intermediate testing. This is a lab, but also an exploration activity.

Begin by creating a standard forms application with the following appearance:



```
02-1E-43-2B-50-2C-43-67-76-24-F5-E4-17-0B-CE-33 : C:\Users\simonw\Desktop\npp.6.9.1.bin\plugins\APIs\javascript.xml
02-E5-B8-29-05-9D-4A-28-17-17-25-D4-45-29-A7-50 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\chineseSimplified.xml
04-E3-E2-7D-9D-63-5C-6B-23-C8-E4-0D-CB-BD-D4-42 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\tagalog.xml
05-6B-92-B4-D2-E1-69-84-50-59-90-CF-37-9B-54-86 : C:\Users\simonw\Desktop\npp.6.9.1.bin\plugins\APIs\sql.xml
07-00-74-35-AD-7A-B2-F7-52-D0-A7-B1-A7-E3-93-2B : C:\Users\simonw\Desktop\npp.6.9.1.bin\themes\Zenburn.xml
07-B1-A1-B6-34-4D-99-B6-20-13-15-86-A2-00-E9-D3 : C:\Users\simonw\Desktop\npp.6.9.1.bin\plugins\APIs\php.xml
0A-68-43-74-49-CF-A3-5F-F6-C0-00-72-6E-7E-99-E9 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\urdu.xml
0E-6C-30-56-E1-48-C3-54-D9-6C-DE-94-39-2F-5B-FD : C:\Users\simonw\Desktop\npp.6.9.1.bin\plugins\APIs\html.xml
0E-DC-3C-4A-40-D4-F4-00-DF-93-03-D3-E8-74-98-34 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\turkish.xml
0F-8F-06-4A-C5-1D-3A-97-C8-47-D8-1C-67-71-9B-2E : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\uyghur.xml
11-4C-38-86-F0-E8-93-18-85-59-DA-CA-5B-03-79-69 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\danish.xml
12-B9-72-B6-91-30-E6-64-E5-0A-C1-11-D2-98-37-9B : C:\Users\simonw\Desktop\npp.6.9.1.bin\plugins\APIs\rc.xml
13-94-E1-E9-D9-12-80-C0-77-A6-15-5B-41-D1-B3-4B : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\vietnamese.xml
14-74-69-21-E7-FD-AF-9B-FA-07-B9-95-9F-9C-B6-C7 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\tamil.xml
14-95-90-CE-BD-70-0A-A8-76-1C-EF-6D-B7-BC-03-7C : C:\Users\simonw\Desktop\npp.6.9.1.bin\stylers.model.xml
14-95-90-CE-BD-70-0A-A8-76-1C-EF-6D-B7-BC-03-7C : C:\Users\simonw\Desktop\npp.6.9.1.bin\stylers.xml
17-4E-7D-C3-67-FF-1C-21-34-32-B8-91-F1-1D-25-BB : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\serbian.xml
18-47-60-0C-7E-DD-6C-EB-C6-DE-2A-E1-BF-FA-A8-9F : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\tajikCyrillic.xml
19-1C-C6-B7-ED-37-FA-D2-74-F9-85-D7-32-9B-D0-48 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\galician.xml
1A-D9-DB-4C-B6-52-9F-10-7B-AB-FC-36-E6-99-2A-8B : C:\Users\simonw\Desktop\npp.6.9.1.bin\session.xml
1B-06-F5-8D-75-60-0B-98-66-4D-2F-E6-E7-9F-08-44 : C:\Users\simonw\Desktop\npp.6.9.1.bin\themes\Black board.xml
1B-A0-7E-9E-9C-7C-CB-F0-95-FE-E8-C2-48-37-55-27 : C:\Users\simonw\Desktop\npp.6.9.1.bin\plugins\APIs\vhdl.xml
1F-9B-54-85-0A-5D-7D-34-DE-D2-F7-91-1A-72-21-75 : C:\Users\simonw\Desktop\npp.6.9.1.bin\plugins\mimeTools.dll
21-48-78-D4-C9-04-9C-87-B8-84-F4-4F-05-1E-51-A1 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\serbianCyrillic.xml
26-51-6F-5A-83-6B-93-C7-55-8D-90-42-DB-2B-4F-51 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\italian.xml
27-8B-86-50-32-71-1F-50-B4-FE-6E-6F-3D-4A-5A-00 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\ligurian.xml
2D-9D-E5-6F-21-2E-FC-DB-FC-52-AC-A7-FF-44-B7-C8 : C:\Users\simonw\Desktop\npp.6.9.1.bin\contextMenu.xml
2F-50-3A-65-6F-68-03-D8-0B-0F-5B-10-D0-97-03-5F : C:\Users\simonw\Desktop\npp.6.9.1.bin\plugins\APIs\cs.xml
31-0A-E7-1E-55-4B-99-A4-C7-1B-54-60-97-DE-55-C0 : C:\Users\simonw\Desktop\npp.6.9.1.bin\plugins\APIs\lisp.xml
31-54-15-96-15-36-B5-45-7F-61-13-53-14-52-7F-7F : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\korean.xml
31-58-E1-0E-8A-9B-3C-0E-84-A7-70-F5-F1-1A-AA-1F : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\malay.xml
33-3A-18-AC-B9-3B-A0-83-E8-66-79-C0-65-B6-9D-15 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\aranese.xml
3C-86-77-87-C5-90-FE-7C-9F-2E-6A-F8-96-02-CC-F6 : C:\Users\simonw\Desktop\npp.6.9.1.bin\localization\hungarian.xml
```

Note: The big control is a **ListBox**, with the font set to fixed-width (*Consolas* in this case). The 'Drop files here!' control is a **Label**, with the **BackColor** set to blue, **AutoSize false**, and the text fully centered. It also has been given a single border. Ensure that anchoring permits making the **ListBox** wider and taller.

Part A – Dragging and Dropping Folders

This lab requires that you drag a folder, or folders, onto the blue label, so ensure that you have read the *Drag and Drop Primer*, included with the lab.

Because we will be interested in getting the names of one or more folder, we will want to receive a collection of dropped filenames (folders are considered files for all intents and purposes in Windows). Because of this, your **DragDrop** handler will look more like this, than the example in the primer:

```
List<string> files = new List<string> ((string[])e.Data.GetData (DataFormats.FileDrop));
```

Each file that you get from the drop list will need to be tested to see if it is a folder. The intent is to enumerate all the files within the folder, so a file won't do. The static **GetAttributes** method in the **File** type can tell you many things about a file, including if it is a directory or not. The **File** type is available in the **System.IO** namespace.

For testing, drop a folder on the label and simply dump the collection to the console. This will validate that you are performing a drag and drop operation correctly.

Next, use the **File.GetAttributes** method, along with the **HasFlag** method to determine if each dropped file is a directory.

This would look something like this, where '**s**' is a file path from the drag and drop operation:

```
if (File.GetAttributes(s).HasFlag(FileAttributes.Directory))
```

You only want to process directories, so if the dropped item is not a directory, you can just ignore it.

Now that you know the item you are looking at is a directory, you can use the **DirectoryInfo** type to operate the folder.

The constructor for **DirectoryInfo** takes a file path to a directory, so creating one looks like this, where '**s**' is a directory path:

```
DirectoryInfo di = new DirectoryInfo(s);
```

You may use the **EnumerateFiles** method of **DirectoryInfo** to search for files within a directory. We want to search for all files, and into all subfolders. To do this, you use the '***.***' filter, and the **AllDirectories** option. This will return a collection of **FileInfo** that represent every file that matches the search term within the directory:

```
List<FileInfo> subs = di.EnumerateFiles("*.*", SearchOption.AllDirectories).ToList();
```

FileInfo has many useful methods and properties, among them, **FullName** being the full path for the file.

As a test, you should drop a folder onto your label, and ensure that you can iterate all the files in the directory, dumping their full path to the console. This will validate that you are ready for the next part.

Part B – Generating Hashes for Files

The overall goal of this lab is to identify files that have the same contents. Comparing the contents of every file to every other file in a directory would be memory and I/O intensive, so it would be great if we could identify files that *might be* the same, and ignore ones that couldn't be.

We could generate a hash for each file. A hash in this case, will be a fixed size (normally small) block of data that is generated from some other data, in our case the contents of the file. Different file contents will generate a different hash value. The exact same contents will generate exactly the same hash. Because the hash data size is smaller than the source data size, it is possible that different source data patterns could produce the same hash. This is very unlikely, but it means that hash values can only be used for *differentiation*. If hash values are the same, it means that the source data *could be* the same. If hash values are different, the source data *is* different.

There are many different hashing algorithms available, and we will use a simple one: MD5.

The `System.Security.Cryptography` namespace contains the `MD5CryptoServiceProvider` type. Create one of these at class-level:

```
private MD5CryptoServiceProvider crypto = new MD5CryptoServiceProvider();
```

Use the form constructor or `Load` event to `Initialize` it:

```
crypto.Initialize();
```

The `MD5CryptoServiceProvider` instance can be used to generate MD5 hashes for our files. The hash size for an MD5 in this context is 16 bytes. The `ComputeHash` method of the `MD5CryptoServiceProvider` instance will return an array of 16 bytes, and wants an array of source bytes to hash. We can load the entire contents of a file as an array of bytes with `File.ReadAllBytes`, and use this as a source of bytes for the `ComputeHash` method, where 'crypto' is an initialized `MD5CryptoServiceProvider` and 'sub' is a `FileInfo`:

```
byte[] hash = crypto.ComputeHash(File.ReadAllBytes(sub.FullName));
```

An array of 16 bytes is not a very useful variable for performing comparisons, so you should write a method that converts an array of 16 byte values into a `string` in the format:

```
2B-9F-E1-62-FB-67-FA-E3-46-3C-AC-00-E8-31-B9-3D
```

You may use whatever technique you like to do this, but you should use string interpolation to ensure that each `byte` is expressed as hex, with two-digit width and leading zeroes where appropriate. The `string.Join` method can also provide the hyphen concatenation, and the `Select` extension method may be of service to project the collection of bytes into a collection of formatted string for each `byte`.

As a test, you should drag and drop a folder with some test files in it onto your label. You should have your code dump out the file path and the generated hash for the file. Do this before you continue.

Part C – Saving the Data

The data we want to work with will be the full file path and the MD5 hash (formatted as described above). To do this, you should create a structure that contains `string` fields for these items.

You will need a `ToString` override to express the data in the structure as a single `string`. This is a necessary evil, due to how the `ListBox` control operates; it takes an `object` and uses `ToString` to obtain the text for each line in the `Items` collection.

```
private struct SMD5File
{
    public string sPath;
    public string sMD5;

    public override string ToString()
    {
        return $"{sMD5} : {sPath}";
    }
}
```

Create an instance of this structure for every file in the directories that you have dropped and add them to a collection at class-level.

Create a method that clears the **`Items`** in the **`ListBox`** and populates it with the elements of the class-level collection. Adding the **`SMD5File`** instances directly to the **`ListBox`** will allow you to reference them later, when the user clicks on an item in the **`ListBox`**.

You should also use the **`OrderBy`** extension method to order the collection of **`SMD5File`** instances by the MD5 string field. While not particularly useful, it will cluster files with the same MD5 value. If you add files with the exact same contents in the test folder, these will appear together, with the same MD5 value in the **`ListBox`**.

You no longer need to dump the file information out to the console, as you should be able to display it in the **`ListBox`**.

Now would be a great time to do some testing and clean up any loose ends that need tidying.

Part D – Looking for Duplicates

When the user clicks on the 'Find Duplicates' button, you will identify all files that have at least one duplicate MD5 value.

You are free to do this any way you like. The resultant collection must only contain one of each actual file, even if it is an MD5 duplicate with more than one file.

Replace the class-level collection with the duplicate containing collection from this operation – only files with duplicates will be listed in the **`ListBox`**.

It is OK if the list contains no items (no duplicates found).

Part E – Resolving Duplicates

When the use clicks on the 'Resolve Duplicates' button, you will attempt to determine if files with the same hash have the same contents.

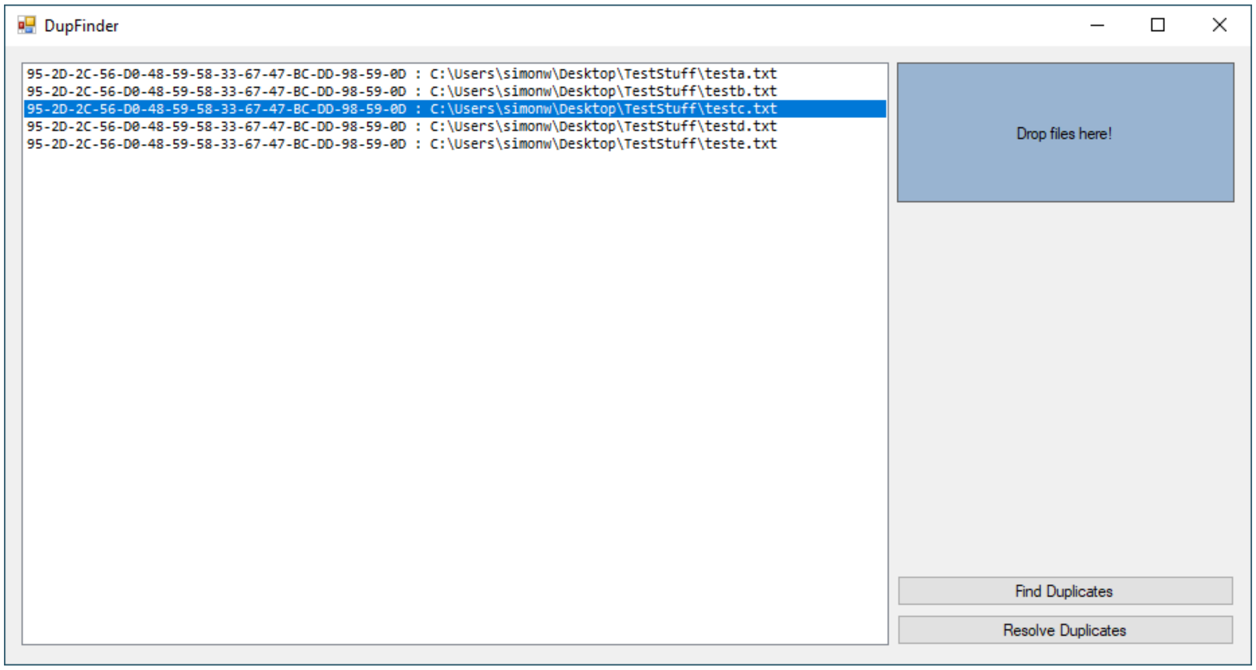
This process will require that you have selected an MD5 in the **`ListBox`**. If an item isn't selected, this button should do nothing.

For the selected item in the **`ListBox`**, you will need to identify all the other files that have the same MD5. You will then need to check every file against every other file to see if the contents are the same. You will need to read the contents of the files once again with **`File.ReadAllBytes`** and compare every byte at each ordinal position to determine if the files are the same. The **`SequenceEqual`** extension method could be of great help here.

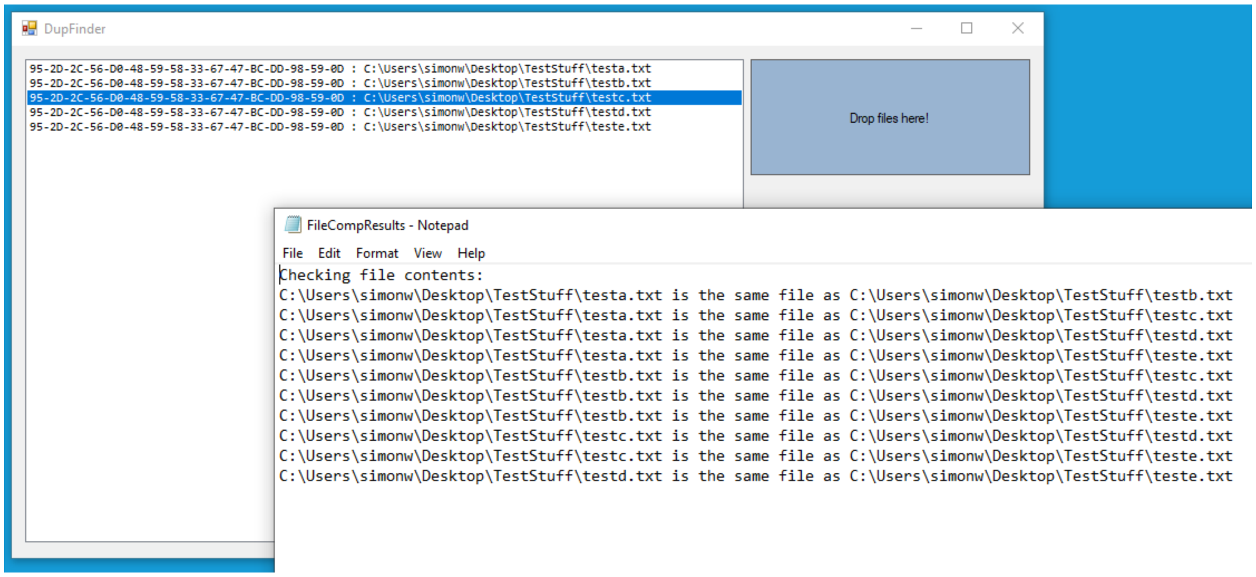
Do not perform redundant file contents to file contents checks here.

For each file-to-file comparison that you perform, append the outcome to a **`string`**. When done, write the string out to a file with **`File.WriteAllText`**, and use `System.Diagnostics.Process.Start` to launch the result in Notepad for viewing.

```
File.WriteAllText("FileCompResults.txt", sResults);
System.Diagnostics.Process.Start("notepad.exe", "FileCompResults.txt");
```



```
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\testa.txt
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\testb.txt
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\testc.txt
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\testd.txt
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\teste.txt
```



```
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\testa.txt
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\testb.txt
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\testc.txt
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\testd.txt
95-2D-2C-56-D0-48-59-58-33-67-47-BC-DD-98-59-0D : C:\Users\simonw\Desktop\TestStuff\teste.txt
```

```
Checking file contents:
C:\Users\simonw\Desktop\TestStuff\testa.txt is the same file as C:\Users\simonw\Desktop\TestStuff\testb.txt
C:\Users\simonw\Desktop\TestStuff\testa.txt is the same file as C:\Users\simonw\Desktop\TestStuff\testc.txt
C:\Users\simonw\Desktop\TestStuff\testa.txt is the same file as C:\Users\simonw\Desktop\TestStuff\testd.txt
C:\Users\simonw\Desktop\TestStuff\testa.txt is the same file as C:\Users\simonw\Desktop\TestStuff\teste.txt
C:\Users\simonw\Desktop\TestStuff\testb.txt is the same file as C:\Users\simonw\Desktop\TestStuff\testc.txt
C:\Users\simonw\Desktop\TestStuff\testb.txt is the same file as C:\Users\simonw\Desktop\TestStuff\testd.txt
C:\Users\simonw\Desktop\TestStuff\testb.txt is the same file as C:\Users\simonw\Desktop\TestStuff\teste.txt
C:\Users\simonw\Desktop\TestStuff\testc.txt is the same file as C:\Users\simonw\Desktop\TestStuff\testd.txt
C:\Users\simonw\Desktop\TestStuff\testc.txt is the same file as C:\Users\simonw\Desktop\TestStuff\teste.txt
C:\Users\simonw\Desktop\TestStuff\testd.txt is the same file as C:\Users\simonw\Desktop\TestStuff\teste.txt
```

Example output from five duplicate files.