---

1. **Create and Delete Directory**
   - **Task**: Write a program that creates a directory at a specified path and then deletes it after confirmation from the user.
   - **Solution**:
    ```csharp
    private void btnCreateDir_Click(object sender, EventArgs e)
    {
        string path = txtDirPath.Text;

        if (!Directory.Exists(path))
        {
            Directory.CreateDirectory(path);
            MessageBox.Show("Directory Created!");
        }
        else
        {
            MessageBox.Show("Directory Already Exists!");
    ```

```
        }
    }

    private void btnDeleteDir_Click(object sender, EventArgs e)
    {
        string path = txtDirPath.Text;

        if (Directory.Exists(path))
        {
            Directory.Delete(path);
            MessageBox.Show("Directory Deleted!");
        }
        else
        {
            MessageBox.Show("Directory Not Found!");
        }
    }
```

2. **List All Files in a Directory**

  - **Task**: Create an application that lists all files in a specified directory.
  - **Solution**:
    ```csharp
    private void btnListFiles_Click(object sender, EventArgs e)
    {
        string path = txtDirPath.Text;

        if (Directory.Exists(path))
        {
            lstFiles.Items.Clear();
            string[] files = Directory.GetFiles(path);
            foreach (var file in files)
            {
                lstFiles.Items.Add(Path.GetFileName(file));
            }
        }
        else
        {
            MessageBox.Show("Directory Not Found!");
    ```

        }
    }

3. **Create and Write to a Text File**
   - **Task**: Develop a program that creates a text file in a specified directory and writes user input to it.
   - **Solution**:
    ```csharp
    private void btnWriteFile_Click(object sender, EventArgs e)
    {
        string path = Path.Combine(txtDirPath.Text, "example.txt");

        File.WriteAllText(path, txtContent.Text);
        MessageBox.Show("File Created and Written!");
    }
    ```

4. **Copy and Move Files**

  - **Task**: Implement functionality to copy and move files between directories.

  - **Solution**:

   ```csharp
   private void btnCopyFile_Click(object sender, EventArgs e)
   {
       string sourcePath = txtSourcePath.Text;
       string destPath = txtDestPath.Text;

       if (File.Exists(sourcePath))
       {
           File.Copy(sourcePath, destPath);
           MessageBox.Show("File Copied!");
       }
       else
       {
           MessageBox.Show("Source File Not Found!");
       }
   }
   ```

```
    private void btnMoveFile_Click(object sender, EventArgs e)
    {
        string sourcePath = txtSourcePath.Text;
        string destPath = txtDestPath.Text;

        if (File.Exists(sourcePath))
        {
            File.Move(sourcePath, destPath);
            MessageBox.Show("File Moved!");
        }
        else
        {
            MessageBox.Show("Source File Not Found!");
        }
    }
```

5. **Read a Text File**

- **Task**: Create an application that reads the contents of a text file and displays it in a text box.

  - **Solution**:
    ```csharp
    private void btnReadFile_Click(object sender, EventArgs e)
    {
        string path = txtFilePath.Text;

        if (File.Exists(path))
        {
            txtFileContents.Text = File.ReadAllText(path);
        }
        else
        {
            MessageBox.Show("File Not Found!");
        }
    }
    ```

6. **Create a Directory Tree Viewer**

   - **Task**: Build a program that recursively lists all directories and files within a specified directory in a tree view control.

   - **Solution**:

    ```csharp
    private void btnLoadTree_Click(object sender, EventArgs e)
    {
        string path = txtDirPath.Text;
        treeView1.Nodes.Clear();

        if (Directory.Exists(path))
        {
            TreeNode rootNode = new TreeNode(Path.GetFileName(path));
            treeView1.Nodes.Add(rootNode);
            LoadDirectory(path, rootNode);
        }
        else
        {
            MessageBox.Show("Directory Not Found!");
        }
    ```

```
    }

    private void LoadDirectory(string path, TreeNode
node)
    {
        string[] subDirs =
Directory.GetDirectories(path);
        foreach (var dir in subDirs)
        {
            TreeNode dirNode = new
TreeNode(Path.GetFileName(dir));
            node.Nodes.Add(dirNode);
            LoadDirectory(dir, dirNode);
        }

        string[] files = Directory.GetFiles(path);
        foreach (var file in files)
        {
            node.Nodes.Add(new
TreeNode(Path.GetFileName(file)));
        }
    }
```

~~~

7. **File Information Viewer**
   - **Task**: Create an application that displays detailed information about a selected file (size, creation date, etc.).
   - **Solution**:
    ```csharp
    private void btnGetFileInfo_Click(object sender, EventArgs e)
    {
        string path = txtFilePath.Text;

        if (File.Exists(path))
        {
            FileInfo fileInfo = new FileInfo(path);
            lblFileInfo.Text = $"File Size: {fileInfo.Length} bytes\n" +
                               $"Created: {fileInfo.CreationTime}\n" +
                               $"Last Accessed: {fileInfo.LastAccessTime}\n" +
~~~

                        $"Last Modified:
{fileInfo.LastWriteTime}";
        }
        else
        {
            MessageBox.Show("File Not Found!");
        }
    }
    ~~~

### 8. **Path Manipulation**
   - **Task**: Write a program that manipulates paths to extract file name, extension, and directory name from a given path.
   - **Solution**:
    ```csharp
    private void btnManipulatePath_Click(object sender, EventArgs e)
    {
        string path = txtFilePath.Text;
    ~~~

        lblFileName.Text = $"File Name: {Path.GetFileName(path)}";
        lblFileExtension.Text = $"File Extension: {Path.GetExtension(path)}";
        lblDirectoryName.Text = $"Directory Name: {Path.GetDirectoryName(path)}";
    }
    ~~~

### 9. **Working with MemoryStream**

   - **Task**: Create an application that writes data to a `MemoryStream` and then reads it back to display in a text box.
   - **Solution**:
    ```csharp
    private void btnWriteMemory_Click(object sender, EventArgs e)
    {
        string text = txtInput.Text;

        using (MemoryStream memoryStream = new MemoryStream())
    ~~~

```
        {
            StreamWriter writer = new StreamWriter(memoryStream);
            writer.Write(text);
            writer.Flush();

            memoryStream.Position = 0;
            StreamReader reader = new StreamReader(memoryStream);
            txtOutput.Text = reader.ReadToEnd();
        }
    }
```

10. **Convert Image to Byte Array and Back Using MemoryStream**

- **Task**: Write a program that loads an image, converts it to a byte array using `MemoryStream`, and then displays the image again.
- **Solution**:

```csharp
```

```
    private void btnLoadImage_Click(object sender,
EventArgs e)
    {
        OpenFileDialog openFileDialog = new
OpenFileDialog();
        if (openFileDialog.ShowDialog() ==
DialogResult.OK)
        {
            byte[] imageBytes;
            using (MemoryStream memoryStream = new
MemoryStream())
            {
                Image image =
Image.FromFile(openFileDialog.FileName);
                image.Save(memoryStream,
image.RawFormat);
                imageBytes = memoryStream.ToArray();
            }

            using (MemoryStream memoryStream = new
MemoryStream(imageBytes))
            {
                pictureBox1.Image = Image.FromStream(memoryStream);
            }
        }
    }
```
