---

1. **Reading from and Writing to a File using FileStream**

  - **Task**: Create a program that reads a text file using `FileStream` and writes its content to another file.

  - **Solution**:

    ```csharp
    private void ReadAndWriteFile()
    {
        using (FileStream fsRead = new FileStream("input.txt", FileMode.Open))
        using (FileStream fsWrite = new FileStream("output.txt", FileMode.Create))
        {
            int data;
            while ((data = fsRead.ReadByte()) != -1)
            {
                fsWrite.WriteByte((byte)data);
            }
        }
    ```

     }
     ~~~

2. **Reading and Writing Binary Data**
   - **Task**: Use `BinaryWriter` to write some primitive data types to a binary file and then use `BinaryReader` to read the data back.
   - **Solution**:
     ```csharp
     private void WriteBinaryData()
     {
         using (BinaryWriter writer = new BinaryWriter(File.Open("data.bin", FileMode.Create)))
         {
             writer.Write(1.25);
             writer.Write(42);
             writer.Write("Hello World");
         }
     }

     private void ReadBinaryData()
     ~~~

    {
        using (BinaryReader reader = new BinaryReader(File.Open("data.bin", FileMode.Open)))
        {
            double a = reader.ReadDouble();
            int b = reader.ReadInt32();
            string c = reader.ReadString();
            MessageBox.Show($"{a}, {b}, {c}");
        }
    }
    ~~~

### 3. **Copying a File using Standard IO Streams**
   - **Task**: Create a program that copies a file using `FileStream` and `BufferedStream` to improve performance.
   - **Solution**:
    ```csharp
    private void CopyFile()
    {
    ~~~

```
        using (FileStream source = new FileStream("source.txt", FileMode.Open))
        using (FileStream destination = new FileStream("destination.txt", FileMode.Create))
        using (BufferedStream buffer = new BufferedStream(destination))
        {
            source.CopyTo(buffer);
        }
    }
    ```

4. **Reading from Standard Input and Writing to Standard Output**

   - **Task**: Write a C# console application that reads a line of text from the console and writes it back in uppercase.

   - **Solution**:

    ```csharp
    static void Main(string[] args)
    {
        Console.WriteLine("Enter a line of text:");
```

```
        string input = Console.ReadLine();
        Console.WriteLine("Uppercase: " + input.ToUpper());
    }
```

5. **Reading and Writing Text Files using StreamReader and StreamWriter**

- **Task**: Use `StreamReader` to read the contents of a text file and `StreamWriter` to write the reversed contents to another file.
- **Solution**:

```csharp
    private void ReverseTextFile()
    {
        using (StreamReader reader = new StreamReader("input.txt"))
        using (StreamWriter writer = new StreamWriter("output.txt"))
        {
            string content = reader.ReadToEnd();
            char[] charArray = content.ToCharArray();
```

            Array.Reverse(charArray);
            writer.Write(charArray);
        }
    }
    ~~~

6. **Counting Words in a Text File using TextReader**

   - **Task**: Create a program that reads a text file using `TextReader` and counts the number of words in the file.

   - **Solution**:

    ```csharp
    private void CountWordsInFile()
    {
        using (TextReader reader = new StreamReader("input.txt"))
        {
            string content = reader.ReadToEnd();
            int wordCount = content.Split(' ', '\n', '\r').Length;
    ~~~

```
            MessageBox.Show($"Word Count: {wordCount}");
        }
    }
```

7. **Writing a Log File using TextWriter**

   - **Task**: Write a log entry to a text file using `TextWriter` each time an event (e.g., button click) occurs.

   - **Solution**:

```csharp
    private void LogEvent(string message)
    {
        using (TextWriter writer = new StreamWriter("log.txt", true))
        {
            writer.WriteLine($"{DateTime.Now}: {message}");
        }
    }
```

    private void btnClick_Click(object sender, EventArgs e)
    {
        LogEvent("Button was clicked.");
    }
    ~~~

8. **Creating a Custom Stream Class**
   - **Task**: Implement a custom stream that counts the number of bytes read or written and use it in a file copy operation.
   - **Solution**:
    ```csharp
    public class CountingStream : Stream
    {
        private Stream _innerStream;
        public long BytesRead { get; private set; }
        public long BytesWritten { get; private set; }

        public CountingStream(Stream innerStream)
        {
    ~~~

```
        _innerStream = innerStream;
    }

    public override int Read(byte[] buffer, int offset,
int count)
    {
        int bytesRead = _innerStream.Read(buffer,
offset, count);
        BytesRead += bytesRead;
        return bytesRead;
    }

    public override void Write(byte[] buffer, int
offset, int count)
    {
        _innerStream.Write(buffer, offset, count);
        BytesWritten += count;
    }

    // Other required Stream methods would
delegate to _innerStream...
```

```
        public override bool CanRead =>
_innerStream.CanRead;
        public override bool CanSeek =>
_innerStream.CanSeek;
        public override bool CanWrite =>
_innerStream.CanWrite;
        public override long Length =>
_innerStream.Length;

        public override long Position
        {
            get => _innerStream.Position;
            set => _innerStream.Position = value;
        }

        public override void Flush() =>
_innerStream.Flush();
        public override long Seek(long offset,
SeekOrigin origin) => _innerStream.Seek(offset,
origin);
        public override void SetLength(long value) =>
_innerStream.SetLength(value);
    }
```

9. **Handling End-of-File (EOF) in StreamReader**

   - **Task**: Create a program that reads a file line by line using `StreamReader` until the end of the file is reached.

   - **Solution**:

    ```csharp
    private void ReadUntilEOF()
    {
        using (StreamReader reader = new StreamReader("input.txt"))
        {
            string line;
            while ((line = reader.ReadLine()) != null)
            {
                Console.WriteLine(line);
            }
        }
    }
    ```

10. **Serializing and Deserializing Objects using FileStream**

   - **Task**: Serialize an object to a binary file using `BinaryWriter` and deserialize it using `BinaryReader`.

   - **Solution**:

     ```csharp
     [Serializable]
     public class Person
     {
         public string Name { get; set; }
         public int Age { get; set; }
     }

     private void SerializePerson(Person person)
     {
         using (FileStream fs = new FileStream("person.dat", FileMode.Create))
         using (BinaryWriter writer = new BinaryWriter(fs))
         {
     ```

```
            writer.Write(person.Name);
            writer.Write(person.Age);
        }
    }

    private Person DeserializePerson()
    {
        using (FileStream fs = new FileStream("person.dat", FileMode.Open))
        using (BinaryReader reader = new BinaryReader(fs))
        {
            string name = reader.ReadString();
            int age = reader.ReadInt32();
            return new Person { Name = name, Age = age };
        }
    }
```
