Section 1

Question 1
Tag Checker Problem
Answer
Please open the project name as **"HtmlChecker"**

Section 2

Question
Provide a concise functional overview of the code excerpt mentioned below,
Answer
Please open the project name as **"Section2Solution"**

Section 3

Question 1
Develop a recursive function that iterates through a directory and executes an arbitrary action on each identified file
Answer
Please open project **" Section3Question1"**

Question 2
Design a database to keep track of student's subjects and marks. Each student will be enrolled in 1 or more courses and will have written 0 or more tests per course. The data provided will be:
Student number
Student Name
Course Number
Course Name
Test date
Test Mark

Answer
Please open the **"DB diagram.jpg"** file to view the diagram drawn for the given information and below tables are given with sample data.

tbl_Students

Id	Number	Name
1	10-16235-1	A
2	10-16233-1	B

3	10-16322-1	C

tbl_Courses

Id	Number	Name
1	CS101	C. F
2	CS102	PL 1
3	CS103	PL 2

tbl_StudentCourses

Id	StudentId	CourseId
1	1	1
2	1	3
3	2	2
4	2	1
5	2	3
6	3	1

tbl_StudentCourseTests

id	StudentCouresId	TestDate	TestMark
1	1	23/05/2020	70
2	3	25/05/2020	80
3	4	27/05/2020	60

Section 4

Question 1

What is shared memory and when will you use it?

Answer

Shared memory for software is a way for different programs to communicate and pass data without more overhead from communications processes. With shared memory, one program writes to the shared memory any data it needs another program to receive.

For example, if Program A wants to give a list to Program B, it saves the data in shared memory and marks it with a semaphore or other flagging system to signal that it is ready to be read by Program B.

When Program B finds the file, it checks the semaphore to see if it is allowed to touch that file. If allowed, then it does what it needs to do to the file, puts it in shared memory or updates it. It also updates the semaphore, so that Program A knows that it should take the file.

Question 2

What are the restrictions, if any, in C# OOP?

Answer

C# does not support multiple-inheritance. But you can use Interfaces.

Question 3

Give an example, in your own words, of polymorphism and when it should be used

Answer:

The word “poly” means many and “morphs” means forms, So it means many forms. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations.

The main benefit of polymorphism is that it simplifies the programming interface. It permits conventions to be established that can be reused in class after class

Question 4

Please provide a practical example of interface implementation in C#, describing exactly what benefits you would achieve through your design

Answer:

By using interfaces it can achieve multiple inheritance. It is also used to achieve loose coupling and also accessing objects using interfaces, implementation hiding, and extensibility. Interfaces are used to implement abstraction.

Practical example given bellow

Home Controller class

```
public class HomeController : ControllerBase
  {
     IHomeService _homeService;

     public HomeController(IHomeService homeService) {
        _homeService = homeService;
     }
```

```
        [HttpGet]
        public ActionResult Get()
        {
            ReadJsonDto readJsonDto = _homeService.ReadJsonFIle();
            if (readJsonDto !=null)
            {
                return Ok(readJsonDto);
            }
            else
            {
                return NotFound();

            }
        }

        [HttpPost]
        public IActionResult Post([FromBody] ReadJsonDto readJsonDto)
        {
          Boolean result = _homeService.save(readJsonDto);

            if (result)
            {
                return Ok(readJsonDto);
            }
            else
            {
                return NotFound();

            }
        }
    }
}
```

Home interface Service

```
    public interface IHomeService
    {
        ReadJsonDto ReadJsonFIle();
        Boolean save(ReadJsonDto readJsonDto);
    }
```

Home Service

```
public class HomeService : IHomeService
  {
     public readonly string filePath = @"D:\test\file\jsonFileSaving.json";
     public ReadJsonDto ReadJsonFIle()
     {
        ReadJsonDto readJsonDto = new ReadJsonDto();
        try {
           if (File.Exists(filePath))
           {
              string file = File.ReadAllText(filePath);
              readJsonDto = Newtonsoft.Json.JsonConvert.DeserializeObject<ReadJsonDto>(file);
           }
           else
           {
              readJsonDto = null;
           }
        }catch (Exception ex)
        {
           readJsonDto = null;
        }
        return readJsonDto;
     }

 public Boolean ReadJsonFIleById(int id)
     {
        Console.WriteLine("hello world " +id);
         return true;
     }

     public Boolean save(ReadJsonDto readJsonDto)
     {
        try
        {
           var jsonFormattedContent =
Newtonsoft.Json.JsonConvert.SerializeObject(readJsonDto);

           if (File.Exists(filePath) == false)
           {
              File.WriteAllText(filePath, jsonFormattedContent);
           }
           else
           {
              File.Delete(filePath);
```

```
                File.WriteAllText(filePath, jsonFormattedContent);
            }
            return true;
        }
        catch (Exception ex)
        {
            return false;
        }
    }

}
```