EX.1:

```
// EXERCISE 1
// Create generic class to represent a generic List and create method to add elements
// and another to display the first element. Make meaningful execution in main.
2 references
public class GenericList<T>
{
    private List<T> _items = new List<T>();

    2 references
    public void Add(T item)
    {
        _items.Add(item);
    }

    1 reference
    public T GetFirst()
    {
        if (_items.Count > 0)
            return _items[0];
        throw new InvalidOperationException("The list is empty.");
    }
}
```

EX.2

```
// EXERCISE 2
// Append what you read from text document 1 to text document 2
// and calculate total numbers of words in text document 2 using lambda expression or LINQ.
1 reference
public static class TextFileOperations
{
    1 reference
    public static void AppendTextAndCountWords(string sourceFile, string destinationFile)
    {
        // Read from source file
        string text1 = File.Exists(sourceFile) ? File.ReadAllText(sourceFile) : "";

        // Read from destination file
        string text2 = File.Exists(destinationFile) ? File.ReadAllText(destinationFile) : "";

        // Append text1 to text2
        File.WriteAllText(destinationFile, text2 + text1);

        // Count words using LINQ
        string updatedText = File.ReadAllText(destinationFile);
        int wordCount = updatedText.Split(new[] { ' ', '\n', '\r' }, StringSplitOptions.RemoveEmptyEntries).Count();

        Console.WriteLine($"Total number of words in {destinationFile}: {wordCount}");
    }
}
```

EX.3:

```
// EXERCISE 3
// Create a city class to represent the city name and the city region.
// Create a school class to represent the school's name and the space.
// Create a teacher class to represent the teacher's name and teacher Id.
// Display City's schools and school's teachers in organized way using LINQ.
// Display the school information that has the highest number of teachers.
2 references
public class City
{
    2 references
    public string Name { get; set; }
    1 reference
    public string Region { get; set; }
    3 references
    public List<School> Schools { get; set; } = new List<School>();
}

5 references
public class School...

7 references
public class Teacher...

}
```

```
public class City
{
    2 references
    public string Name { get; set; }
    1 reference
    public string Region { get; set; }
    3 references
    public List<School> Schools { get; set; } = new List<School>();
}

5 references
public class School
{
    4 references
    public string Name { get; set; }
    3 references
    public int Space { get; set; }
    5 references
    public List<Teacher> Teachers { get; set; } = new List<Teacher>();
}

7 references
public class Teacher
{
    4 references
    public string Name { get; set; }
    4 references
    public int TeacherId { get; set; }
}
```

Execution Level :

```
        // Exercise 3: City, School, Teacher classes and LINQ operations
        Console.WriteLine("Exercise 3: City, School, Teacher classes and LINQ operations:");
        City city = new City[...];
        Console.WriteLine("\nSchools and Teachers in " + city.Name + ":");
        foreach (var school in city.Schools)
        {
            Console.WriteLine($"School: {school.Name} (Space: {school.Space})");
            foreach (var teacher in school.Teachers)
            {
                Console.WriteLine($"  Teacher: {teacher.Name} (ID: {teacher.TeacherId})");
            }
        }

        var schoolWithMostTeachers = city.Schools
            .OrderByDescending(s => s.Teachers.Count)
            .FirstOrDefault();
        Console.WriteLine($"\nSchool with most teachers: {schoolWithMostTeachers?.Name} (Teachers: {schoolWithMostTeachers?.Teachers.Count})");
        Console.Write("\n");

    }
```

```
    static void Main(string[] args)
    {

        // Exercise 1: Generic class to represent a generic List
        Console.WriteLine("Exercise 1: Generic class to represent a generic List:");
        GenericList<int> intList = new GenericList<int>();
        intList.Add(5);
        intList.Add(10);
        Console.WriteLine("First element: " + intList.GetFirst());
        Console.Write("\n");


        // Exercise 2: Append text from one document to another and count total words
        Console.WriteLine("Exercise 2: Append text from one document to another and count total words:");
        string file1 = "C:\\Users\\96277\\Desktop\\text1.txt";
        string file2 = "C:\\Users\\96277\\Desktop\\text2.txt";
        TextFileOperations.AppendTextAndCountWords(file1, file2);
        Console.Write("\n");
```

Results:

```
Microsoft Visual Studio Debu

Exercise 1: Generic class to represent a generic List:
First element: 5

Exercise 2: Append text from one document to another and count total words:
Total number of words in C:\Users\96277\Desktop\text2.txt: 15

Exercise 3: City, School, Teacher classes and LINQ operations:

Schools and Teachers in Sample City:
School: Central High (Space: 5000)
  Teacher: Mr. Smith (ID: 1)
  Teacher: Ms. Davis (ID: 2)
School: Westwood Elementary (Space: 3000)
  Teacher: Mrs. Thompson (ID: 3)

School with most teachers: Central High (Teachers: 2)


C:\Users\96277\source\repos\stram\bin\Debug\net8.0\stram.exe (process 23408) exited with code 0 (0x0).
Press any key to close this window . . .
```