Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 2 з дисципліни
«Сучасні технології розробки WEB-застосувань на платформі Microsoft.NET»

«Модульне тестування. Ознайомлення з засобами та практиками модульного тестування»

Виконав *ІП-15, Тонконог В.В.* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Бардін В.* _____
(прізвище, ім'я, по батькові)

Київ 2023

Варіант 8

Завдання

8	Кільцевий список	Див. List<T>, LinkedList<T>	Збереження даних за допомогою динамічно зв'язаного списку

Код виконання:

```
public class MyLinkedListTests
{
    [Fact]
    public void CreateMyLinkedList_WithoutParams_EmpltyList()
    {
        //Act
        var actialResult = new MyLinkedList<object>();

        //Assert
        actialResult.Count.Should().Be(0);
        actialResult.Head.Should().BeNull();
        actialResult.Tail.Should().BeNull();
    }

    [Fact]
    public void
CreateMyLinkedList_WithNullList_ShouldThrowArgumentNullException()
    {
        //Act
        var action = () => new MyLinkedList<object>(null);

        //Assert
        action.Should().Throw<ArgumentNullException>();
    }

    [Fact]
    public void Add_NullParam_ThrowArhumentNullException()
    {
        //Arrange
        var list = new MyLinkedList<object>();

        //Act
        var action = () => list.Add(null);

        //Assert
        action.Should().Throw<ArgumentNullException>();
    }

    [Fact]
    public void Add_FirstValueToEmtyList_HeadEqualTailCount1()
    {
        //Arrange
        var list = new MyLinkedList<object>();

        //Act
        list.Add("dd");
```

```
        //Assert
        list.Head.Should().Be(list.Tail);
        list.Head.Value.Should().Be("dd");
        list.Count.Should().Be(1);
    }

    [Fact]
    public void AddToNonEmptyList_SomeValue_CorrectTail()
    {
        //Arrange
        var list = new MyLinkedList<int>(new List<int> { 1, 2, 3 });

        //Act
        list.Add(4);

        //Assert
        list.Head.Previous.Value.Should().Be(4);
        list.Tail.Value.Should().Be(4);
        list.Count.Should().Be(4);
    }

    [Fact]
    public void AddFirst_FirstValueToEmtyList_HeadEqualTailCount1()
    {
        //Arrange
        var list = new MyLinkedList<object>();

        //Act
        list.AddFirst("dd");

        //Assert
        list.Head.Should().Be(list.Tail);
        list.Head.Value.Should().Be("dd");
        list.Count.Should().Be(1);
    }

    [Fact]
    public void AddFirstToNonEmptyList_SomeValue_CorrectHead()
    {
        //Arrange
        var list = new MyLinkedList<int>(new List<int> { 1, 2, 3 });

        //Act
        list.AddFirst(4);

        //Assert
        list.Head.Value.Should().Be(4);
        list.Tail.Next.Value.Should().Be(4);
        list.Count.Should().Be(4);
    }

    [Fact]
    public void AddFirst_NullParam_ThrowArhumentNullException()
    {
        //Arrange
        var list = new MyLinkedList<object>();

        //Act
        var action = () => list.Add(null);

        //Assert
        action.Should().Throw<ArgumentNullException>();
    }
```

```
    [Fact]
    public void Clear_ShouldBeEmtyList()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });

        //Act
        list.Clear();

        //Assert
        list.Head.Should().BeNull();
        list.Tail.Should().BeNull();
        list.Count.Should().Be(0);
    }

    [Fact]
    public void Find_NonContains_Null()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });

        //Act
        var result = list.Find(4);

        //Assert
        result.Should().BeNull();
    }

    [Fact]
    public void Find_Contains_CorrectValue()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });

        //Act
        var result = list.Find(3);

        //Assert
        result.Value.Should().Be(3);
    }

    [Fact]
    public void Remove_NonContainsElement_False()
    {
        //Arrange
        var array = new int[] { 1, 2, 3, };
        var list = new MyLinkedList<int>(array);

        //Act
        var result = list.Remove(4);


        //Assert
        result.Should().Be(false);
        list.ToList().Should().BeEquivalentTo(array);
    }

    [Fact]
    public void Remove_ContainsElement_True()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });

        //Act
```

```
        var result = list.Remove(4);

        //Assert
        result.Should().Be(false);
        list.Count.Should().Be(3);
    }

    [Fact]
    public void RemoveNode_RemoveHead_True()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });
        var newHead = list.Head.Next;

        //Act
        var result = list.Remove(list.Head);

        //Assert
        result.Should().Be(true);
        list.Head.Should().Be(newHead);
        list.Count.Should().Be(2);
        list.ToArray().Should().BeEquivalentTo(new int[] { 2, 3 });
    }

    [Fact]
    public void RemoveNode_RemoveTail_True()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });
        var newTail = list.Tail.Previous;

        //Act
        var result = list.Remove(list.Tail);

        //Assert
        result.Should().Be(true);
        list.Tail.Should().Be(newTail);
        list.Count.Should().Be(2);
        list.ToArray().Should().BeEquivalentTo(new int[] { 1, 2 });
    }

    [Fact]
    public void RemoveNode_RemoveAloneElement_True()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1 });

        //Act
        var result = list.Remove(list.Tail);

        //Assert
        result.Should().Be(true);
        list.Tail.Should().BeNull();
        list.Head.Should().BeNull();
        list.Count.Should().Be(0);
        list.ToArray().Should().BeEquivalentTo(new int[] {});
    }

    [Fact]
    public void RemoveNode_RemoveNonContailsNode_False()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3 });
        var oldHead = list.Head;
```

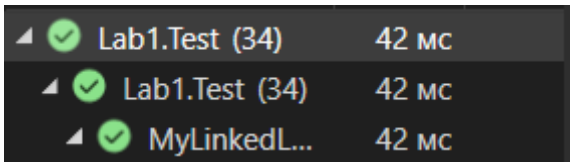
```
        var oldTail = list.Tail;
        var nodeToRemove = new MyLinkedListNode<int>(1);

        //Act
        var result = list.Remove(nodeToRemove);

        //Assert
        result.Should().Be(false);
        list.Tail.Should().Be(oldTail);
        list.Head.Should().Be(oldHead);
        list.Count.Should().Be(3);
        list.ToArray().Should().BeEquivalentTo(new int[] { 1, 2, 3 });
    }

    [Fact]
    public void RemoveNode_Null_ShouldThrowArgumentNullException()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1 });
        var oldHead = list.Head;
        var oldTail = list.Tail;

        //Act
        var action = () => list.Remove(null);

        //Assert
        action.Should().Throw<ArgumentNullException>();
        list.Tail.Should().Be(oldTail);
        list.Head.Should().Be(oldHead);
        list.Count.Should().Be(1);
        list.ToArray().Should().BeEquivalentTo(new int[] { 1 });
    }

    [Fact]
    public void Contains_NonContains_False()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });

        //Act
        var result = list.Contains(4);

        //Assert
        result.Should().BeFalse();
    }

    [Fact]
    public void Contains_ContainsElement_True()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });

        //Act
        var result = list.Contains(3);

        //Assert
        result.Should().BeTrue();
    }

    [Fact]
    public void ContainsNode_ContainsNode_True()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });
```

```
        //Act
        var result = list.Contains(list.Head);

        //Assert
        result.Should().BeTrue();
    }

    [Fact]
    public void ContainsNode_NonContainsNode_Flase()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });

        //Act
        var result = list.Contains(new MyLinkedListNode<int>(1));

        //Assert
        result.Should().BeFalse();
    }

    [Fact]
    public void ContainsNode_NullNode_Flase()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });

        //Act
        var result = list.Contains(null);

        //Assert
        result.Should().BeFalse();
    }

    [Fact]
    public void
CopyTo_ArrayIndexLessThan0_ShouldThrowArgumentOutOfRangeException()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });
        var arr = new int[10];

        //Act
        var action = () => list.CopyTo(arr, -1);

        //Assert
        action.Should().Throw<ArgumentOutOfRangeException>();
    }

    [Fact]
    public void
CopyTo_ArrayIndexMoreThenArrayLength_ShouldThrowArgumentOutOfRangeException()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });
        var arr = new int[10];

        //Act
        var action = () => list.CopyTo(arr, 12);

        //Assert
        action.Should().Throw<ArgumentOutOfRangeException>();
    }
```

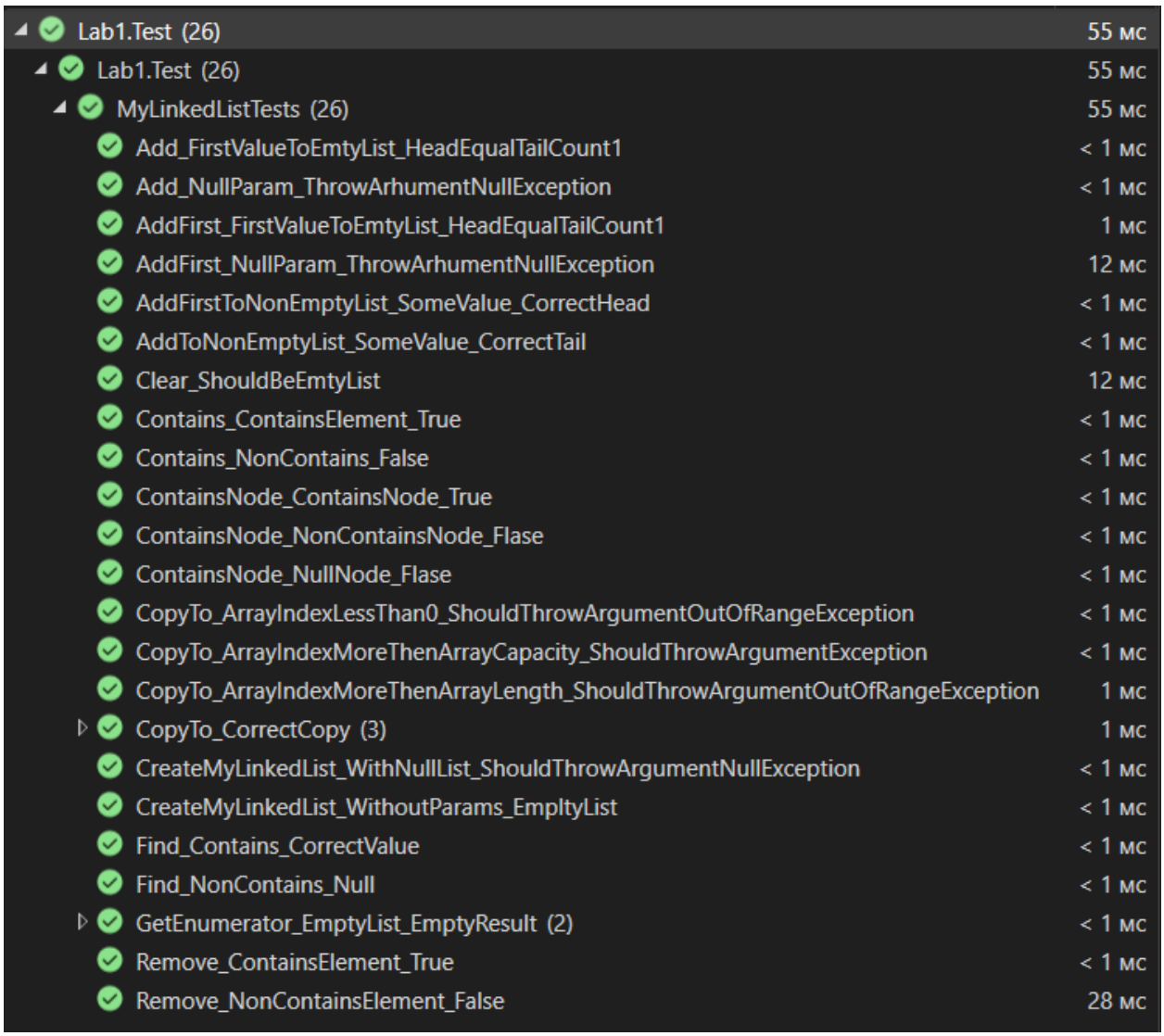
```
    [Fact]
    public void
CopyTo_ArrayIndexMoreThenArrayCapacity_ShouldThrowArgumentException()
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });
        var arr = new int[10];

        //Act
        var action = () => list.CopyTo(arr, 9);

        //Assert
        action.Should().Throw<ArgumentException>();
    }

    [Theory]
    [InlineData(new int[] { 1, 2, 3, 4, 5 }, 2, new int[] { 1, 2, 1, 2, 3 })]
    [InlineData(new int[] { 1, 2, 3, 4, 5 }, 0, new int[] { 1, 2, 3, 4, 5 })]
    [InlineData(new int[] { 1, 2, 3, 4, 5 }, 1, new int[] { 1, 1, 2, 3, 5 })]
    public void CopyTo_CorrectCopy(int[] actualArray, int arayIndex, int[]
expectedResult)
    {
        //Arrange
        var list = new MyLinkedList<int>(new int[] { 1, 2, 3, });

        //Act
        list.CopyTo(actualArray, arayIndex);

        //Assert
        actualArray.Should().BeEquivalentTo(expectedResult);
    }

    [Theory]
    [InlineData(new int[] { 1, 2, 3, 4, 5 }, new int[] { 1, 2, 3, 4, 5 })]
    [InlineData(new int[] { 1, 27, 35 }, new int[] { 1, 27, 35 })]
    public void GetEnumerator_EmptyList_EmptyResult(IEnumerable<int> list,
int[] expectedResult)
    {
        //Arrange
        var myLinkedList = new MyLinkedList<int>(list);

        //Act
        var actualResult = myLinkedList.ToList();

        //Assert
        actualResult.Should().BeEquivalentTo(expectedResult);
    }

    [Fact]
    public void GetHashCode_ValueIsNull_ShouldBeZero()
    {
        //Arrange
        var list = new MyLinkedList<string>(new string[] { "ddd" });
        list.Head.Value = null;

        //Act
        var hashCode = list.Head.GetHashCode();

        //Assert
        hashCode.Should().Be(0);
    }

    [Fact]
    public void GetHashCode_ForEqualLists_ShouldBeEqual()
```

```
    {
        //Arrange
        var listFirst = new MyLinkedList<string>(new string[] { "ddd" });
        var listSecond = new MyLinkedList<string>(new string[] { "ddd" });

        //Act
        var hashCodeFirst = listFirst.Head.GetHashCode();
        var hashCodeSecond = listSecond.Head.GetHashCode();

        //Assert
        hashCodeFirst.Should().Be(hashCodeSecond);
    }

    [Fact]
    public void IsReadOnly_ShouldBeFalse()
    {
        //Arrange
        var list = new MyLinkedList<string>(new string[] { "ddd" });

        //Act
        var isReadOnly = list.IsReadOnly;

        //Assert
        isReadOnly.Should().BeFalse();
    }
}
```

Результат виконання тестів:



Ступінь покриття колекції тестами:


Lab1.Test (26) 55 мс
Lab1.Test (26) 55 мс
MyLinkedListTests (26) 55 мс
Add_FirstValueToEmtyList_HeadEqualTailCount1 < 1 мс
Add_NullParam_ThrowArhumentNullException < 1 мс
AddFirst_FirstValueToEmtyList_HeadEqualTailCount1 1 мс
AddFirst_NullParam_ThrowArhumentNullException 12 мс
AddFirstToNonEmptyList_SomeValue_CorrectHead < 1 мс
AddToNonEmptyList_SomeValue_CorrectTail < 1 мс
Clear_ShouldBeEmtyList 12 мс
Contains_ContainsElement_True < 1 мс
Contains_NonContains_False < 1 мс
ContainsNode_ContainsNode_True < 1 мс
ContainsNode_NonContainsNode_Flase < 1 мс
ContainsNode_NullNode_Flase < 1 мс
CopyTo_ArrayIndexLessThan0_ShouldThrowArgumentOutOfRangeException < 1 мс
CopyTo_ArrayIndexMoreThenArrayCapacity_ShouldThrowArgumentException < 1 мс
CopyTo_ArrayIndexMoreThenArrayLength_ShouldThrowArgumentOutOfRangeException 1 мс
CopyTo_CorrectCopy (3) 1 мс
CreateMyLinkedList_WithNullList_ShouldThrowArgumentNullException < 1 мс
CreateMyLinkedList_WithoutParams_EmpltyList < 1 мс
Find_Contains_CorrectValue < 1 мс
Find_NonContains_Null < 1 мс
GetEnumerator_EmptyList_EmptyResult (2) < 1 мс
Remove_ContainsElement_True < 1 мс
Remove_NonContainsElement_False 28 мс