

Homework: Data Structures, Algorithms and Complexity

This document defines the **homework assignments** for the ["Data Structures" course @ Software University](#). Please submit a single **zip / rar / 7z** archive holding the solutions (source code) of all below described problems.

Class StupidList<T>

You are given the C# class **StupidList<T>** which implements a list of generic type **T** with operations **Add(T)**, **Remove(index)**, **RemoveFirst()**, **RemoveLast()**, **this[index]** (access by index), **Length**, **First** and **Last**:

```
public class StupidList<T>
{
    private T[] arr = new T[0];

    public int Length
    {
        get
        {
            return this.arr.Length;
        }
    }

    public T this[int index]
    {
        get
        {
            return this.arr[index];
        }
    }

    public T First
    {
        get
        {
            return this.arr[0];
        }
    }

    public T Last
    {
        get
        {
            return this.arr[this.arr.Length - 1];
        }
    }

    public void Add(T item)
    {
        var newArr = new T[this.arr.Length + 1];
        Array.Copy(this.arr, newArr, this.arr.Length);
        newArr[newArr.Length - 1] = item;
        this.arr = newArr;
    }
}
```

```

public T Remove(int index)
{
    T result = this.arr[index];
    var newArr = new T[this.arr.Length - 1];
    Array.Copy(this.arr, newArr, index);
    Array.Copy(this.arr, index + 1, newArr, index, this.arr.Length - index - 1);
    this.arr = newArr;
    return result;
}

public T RemoveFirst()
{
    return this.Remove(0);
}

public T RemoveLast()
{
    return this.Remove(this.Length - 1);
}
}

```

Submit the results of all complexity calculations in a single text file.

Problem 1. Add(T) Complexity

Calculate the expected running time $O(f(n))$ of the **Add(T)** operation in the above code in the **worst case**.

Problem 2. Remove(index) Complexity – Worst Case

Calculate the expected running time $O(f(n))$ of the **Remove(index)** operation in the **worst case**.

Problem 3. Remove(index) Complexity – Best Case

Calculate the expected running time $O(f(n))$ of the **Remove(index)** operation in the **best case**.

Problem 4. Remove(index) Complexity – Average Case

Calculate the expected running time $O(f(n))$ of the **Remove(index)** operation in the **average case**.

Problem 5. RemoveFirst(T) Complexity

Calculate the expected running time $O(f(n))$ of the **RemoveFirst(T)** operation. Submit the result in a text file.

Problem 6. RemoveLast(T) Complexity

Calculate the expected running time $O(f(n))$ of the **RemoveLast(T)** operation. Submit the result in a text file.

Problem 7. Length Complexity

Calculate the expected running time $O(f(n))$ of the **Length** operation. Submit the result in a text file.

Problem 8. This[index] Complexity

Calculate the expected running time $O(f(n))$ of the operation **this[index]**. Submit the result in a text file.

Problem 9. First Complexity

Calculate the expected running time $O(f(n))$ of the **First** operation. Submit the result in a text file.

Problem 10. Last Complexity

Calculate the expected running time $O(f(n))$ of the **Last** operation. Submit the result in a text file.