Leet code Problems:

1. https://leetcode.com/problems/minimum-depth-of-binary-tree/

Submission Link: https://leetcode.com/submissions/detail/1244778857/
Code:

```
public class Solution {
    public int MinDepth(TreeNode root) {
        if( root == null)
            return 0;
        if( root.right == null)
            return MinDepth(root.left) + 1;

        if( root.left == null)
            return MinDepth(root.right) + 1;

        return Math.Min(MinDepth(root.left), MinDepth(root.right)) + 1;
    }
}
```

With Task:
Submission Link: https://leetcode.com/submissions/detail/1244854156/

```
public class Solution {
    public int MinDepth(TreeNode root) {
        return MinDepthAsync(root).Result;
    }

    public Task<int> MinDepthAsync(TreeNode root) {
        if (root == null)
            return 0;

        Task<int> leftDepthTask = Task.Run(() => MinDepthAsync(root.left));
        Task<int> rightDepthTask = Task.Run(() => MinDepthAsync(root.right));

        int leftDepth = await leftDepthTask;
        int rightDepth = await rightDepthTask;

        if (root.right == null)
            return leftDepth + 1;

        if (root.left == null)
            return rightDepth + 1;
```

```
        return Math.Min(leftDepth, rightDepth) + 1;
    }
}
```

Problem 2:
Question: https://leetcode.com/problems/excel-sheet-column-title/
Submission: https://leetcode.com/submissions/detail/1244802299/

Code:

```
public class Solution {
    public string ConvertToTitle(int columnNumber) {
        var res = "";
        int baseVal = 65;
        while(columnNumber > 0){
            var tmp = (columnNumber - 1) % 26;
            columnNumber = (columnNumber - 1)/ 26;
            res = (char) (tmp + baseVal) + res;
        }
        return res;
    }
}
```

Async Code:

```
using System.Threading.Tasks;
public class Solution {
    public string ConvertToTitle(int columnNumber) {

        return ConvertToTitleAsync(columnNumber).Result;
    }

    public static async Task<string> ConvertToTitleAsync(int columnNumber)
    {
        return await Task.Run(() =>
        {
            var res = "";
            int baseVal = 65;
            while (columnNumber > 0)
            {
                var tmp = (columnNumber - 1) % 26;
```

```
            columnNumber = (columnNumber - 1) / 26;
            res = (char)(tmp + baseVal) + res;
        }
        return res;
    });
  }
}
```

Problem 3:
Question: https://leetcode.com/problems/linked-list-cycle/
Submission: https://leetcode.com/submissions/detail/1244780970/
Code:

```
public class Solution {
    public bool HasCycle(ListNode head) {
        var set = new HashSet<ListNode>();
        if (head == null)
            return false;

        while(head.next != null){
            if(set.Contains(head))
                return true;
            set.Add(head);
            head = head.next;
        }
        return false;
    }
}
```

Async Code:

```
using System.Threading.Tasks;
public class Solution {
    public bool HasCycle(ListNode head) {

        return CheckForCycleAsync(head).Result;
    }

    private async Task<bool> CheckForCycleAsync(ListNode head) {
        var set = new HashSet<ListNode>();
```

```
        if (head == null)
            return false;

        while (head.next != null) {
            if (set.Contains(head))
                return true;
            set.Add(head);
            head = head.next;
        }
        return false;
    }
}
```