COIS 3020 Assignment 2

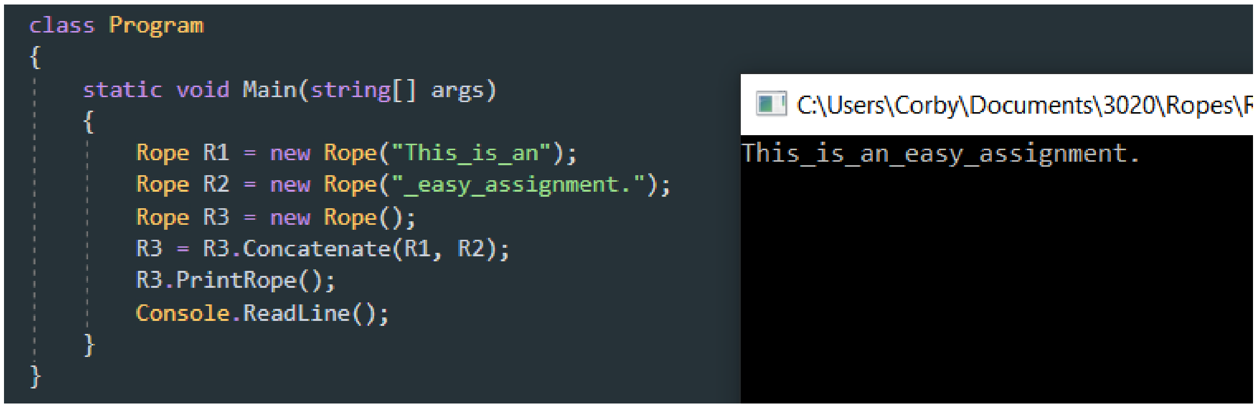
Ropes - Testing

While I tried as best as I could to implement the Rope structure, I ran into one major barrier that prevented the program from going smoothly. I had a lot of trouble implementing the split method where I needed to travel back up the tree and re link the dangling nodes. The split method also puts my tree wildly out of balance, which can also interfere with some other operations. That being said, I made my best effort to try and get most of the methods working. The results are recorded in this document.

private Node Build(string s), public void PrintRope()

This method recursively generates a balanced rope from a given string. IPrint rope prints the rope as a string. These integral method's results can be seen in the following test cases.

public Rope Concatenate(Rope R1, Rope R2)



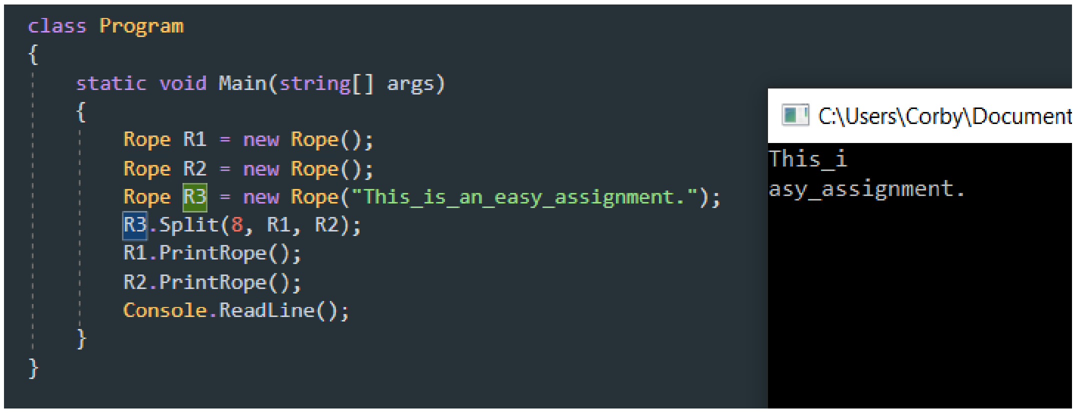
```
class Program
{
    static void Main(string[] args)
    {
        Rope R1 = new Rope("This_is_an");
        Rope R2 = new Rope("_easy_assignment.");
        Rope R3 = new Rope();
        R3 = R3.Concatenate(R1, R2);
        R3.PrintRope();
        Console.ReadLine();
    }
}
```

```
C:\Users\Corby\Documents\3020\Ropes\F
This_is_an_easy_assignment.
```

In the above example, Ropes R1 and R2 are concatenated into Rope R3, which is printed.

private void Split(int i, Rope R1, Rope R2)

This is the method that really gave me trouble throughout this whole assignment. I was able to get the split working insofar as locating the indexed character, and creating two separate ropes. My issue comes up when I try to "hook up" the dangling nodes to the main rope. It often creates the child nodes for the split, and then I am unable to properly recover these characters. I had to do the rest of the assignment with this in mind, as well as the fact that it left my tree wildly unbalanced. Split was made public for the sake of testing.

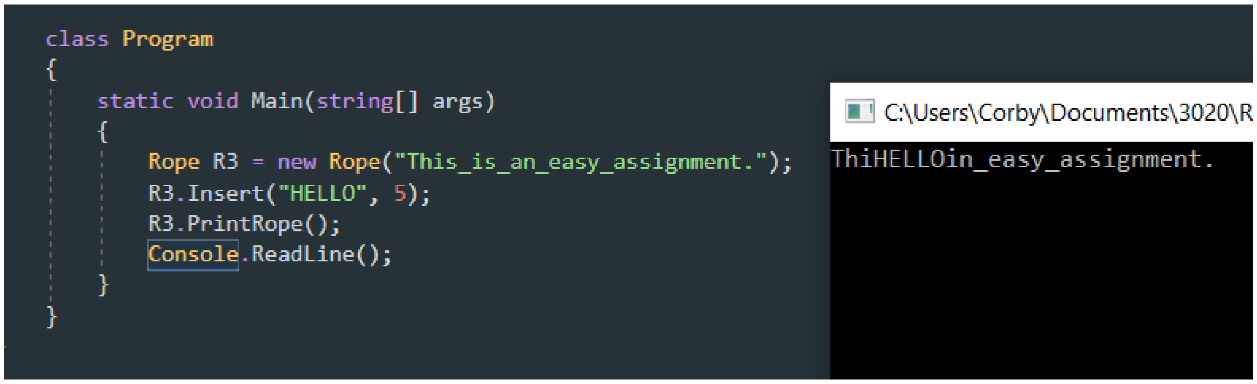


```
class Program
{
    static void Main(string[] args)
    {
        Rope R1 = new Rope();
        Rope R2 = new Rope();
        Rope R3 = new Rope("This_is_an_easy_assignment.");
        R3.Split(8, R1, R2);
        R1.PrintRope();
        R2.PrintRope();
        Console.ReadLine();
    }
}
```

```
C:\Users\Corby\Document
This_i
asy_assignment.
```

public void Insert(string S, int i)

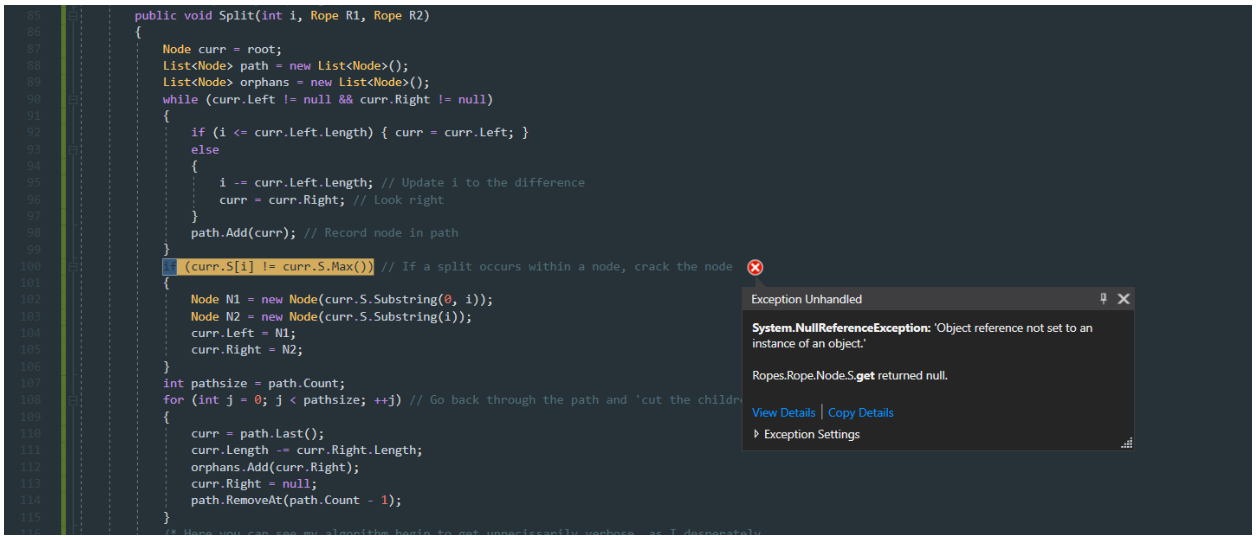
This method inserts text at a given index, and given the limitations of the split method, this one actually works fairly well. The algorithm works the same as discussed in class.



```
class Program
{
    static void Main(string[] args)
    {
        Rope R3 = new Rope("This_is_an_easy_assignment.");
        R3.Insert("HELLO", 5);
        R3.PrintRope();
        Console.ReadLine();
    }
}
```

```
C:\Users\Corby\Documents\3020\R
ThiHELLOin_easy_assignment.
```

public void Delete(int i, int j)



```
public void Split(int i, Rope R1, Rope R2)
{
    Node curr = root;
    List<Node> path = new List<Node>();
    List<Node> orphans = new List<Node>();
    while (curr.Left != null && curr.Right != null)
    {
        if (i <= curr.Left.Length) { curr = curr.Left; }
        else
        {
            i -= curr.Left.Length; // Update i to the difference
            curr = curr.Right; // Look right
        }
        path.Add(curr); // Record node in path
    }
    if (curr.S[i] != curr.S.Max()) // If a split occurs within a node, crack the node
    {
        Node N1 = new Node(curr.S.Substring(0, i));
        Node N2 = new Node(curr.S.Substring(i));
        curr.Left = N1;
        curr.Right = N2;
    }
    int pathsize = path.Count;
    for (int j = 0; j < pathsize; ++j) // Go back through the path and 'cut the childr
    {
        curr = path.Last();
        curr.Length -= curr.Right.Length;
        orphans.Add(curr.Right);
        curr.Right = null;
        path.RemoveAt(path.Count - 1);
    }
    /* Here you can see my algorithm begin to get unnecissarily verbose, as I desperately
```

Exception Unhandled

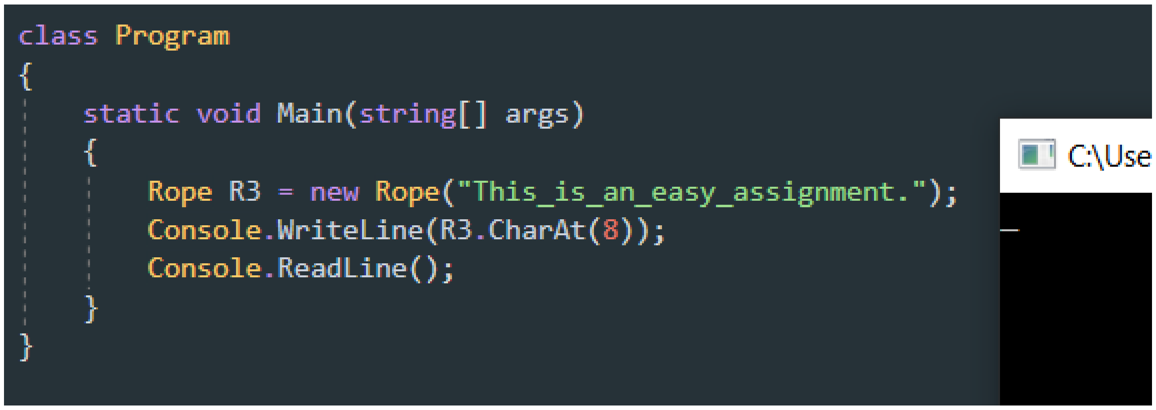
System.NullReferenceException: 'Object reference not set to an instance of an object.'

Ropes.Rope.Node.S.**get** returned null.

While I believe my detete method is coded properly, because it uses two splits, and my the tree is very out of balance after the first one, and I am unable to balance the tree, I cannot run delete again. This also hinders the success of my Substring method.

public char CharAt(int i)

Because ChatAt is essentially the first part of the split, mine worked perfectly :)



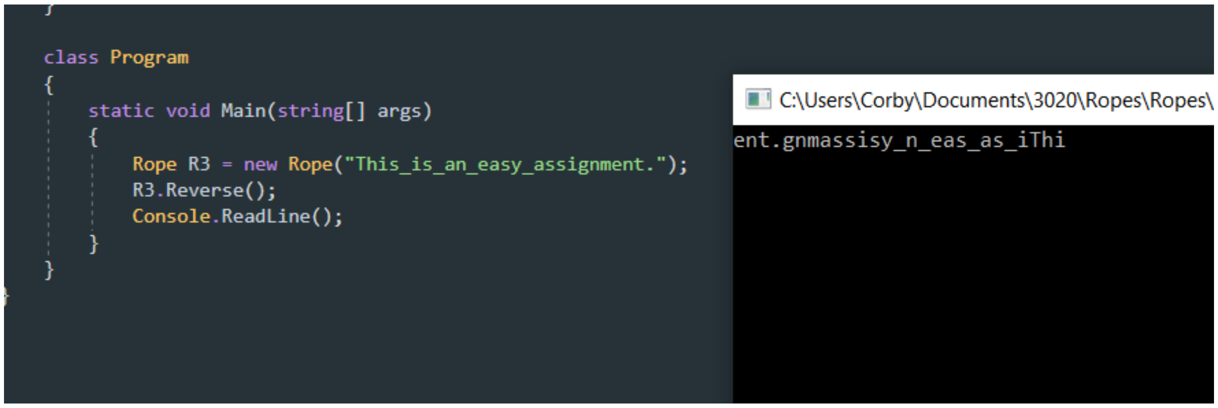
```
class Program
{
    static void Main(string[] args)
    {
        Rope R3 = new Rope("This_is_an_easy_assignment.");
        Console.WriteLine(R3.CharAt(8));
        Console.ReadLine();
    }
}
```

C:\Use

```
_
```

public void Reverse()

I was able to get my reverse to reverse the nodes in the rope, but not the strings in the nodes.

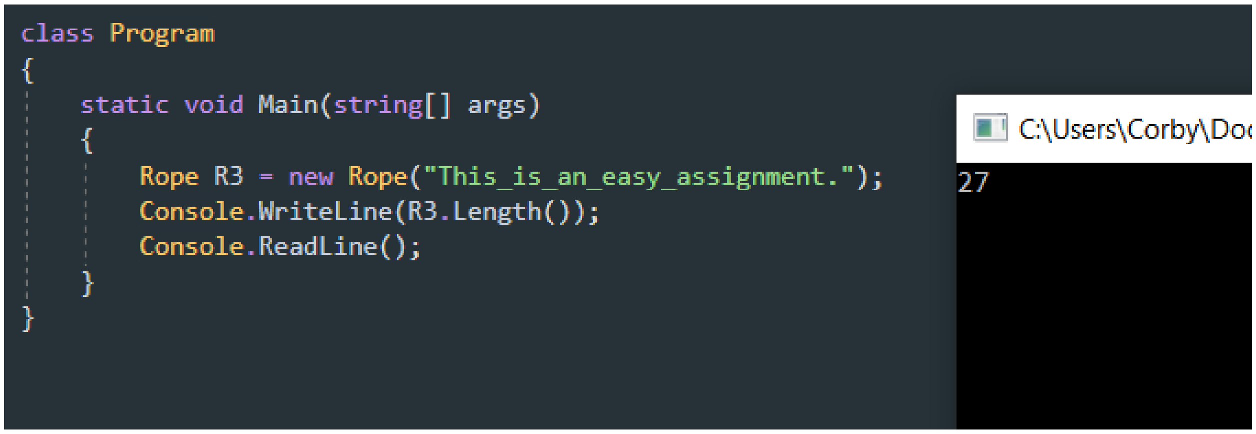


```
class Program
{
    static void Main(string[] args)
    {
        Rope R3 = new Rope("This_is_an_easy_assignment.");
        R3.Reverse();
        Console.ReadLine();
    }
}
```

C:\Users\Corby\Documents\3020\Ropes\Ropes\

```
ent.gnmassisy_n_eas_as_iThi
```

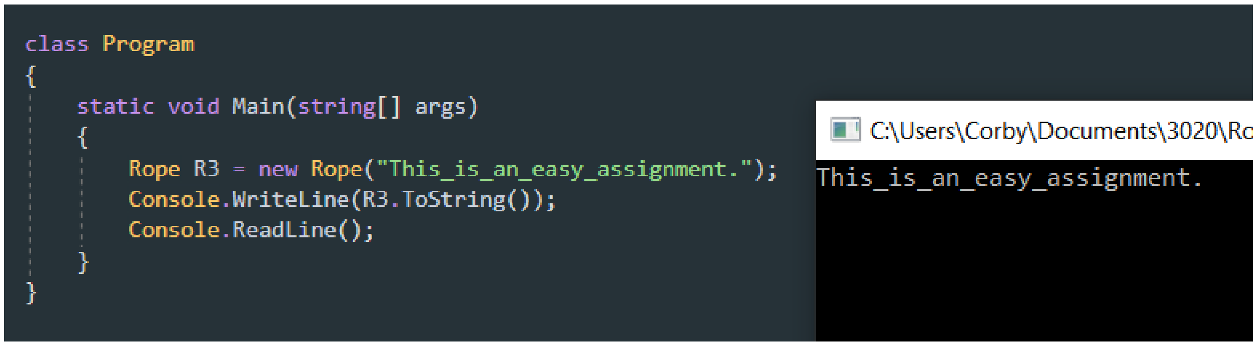
public int Length()



```
class Program
{
    static void Main(string[] args)
    {
        Rope R3 = new Rope("This_is_an_easy_assignment.");
        Console.WriteLine(R3.Length());
        Console.ReadLine();
    }
}
```

```
27
```

public string toString()



```
class Program
{
    static void Main(string[] args)
    {
        Rope R3 = new Rope("This_is_an_easy_assignment.");
        Console.WriteLine(R3.ToString());
        Console.ReadLine();
    }
}
```

```
This_is_an_easy_assignment.
```