Grammar class

```
using System;
using System.Collections.Generic;
using System.Text;

namespace LL1
{
    public class Grammar
    {
        public char[] Terminals { get; }
        public char[] NonTerminals { get; }

        public List<Rule> RuleSet;

        private List<Alternative> Alternatives { get; set; }

        public Grammar(char[] terminals, char[] nonTerminals)
        {
            Terminals = terminals;
            NonTerminals = nonTerminals;
            RuleSet = new List<Rule>();
        }

        public void AddRule(char left, string rigth)
        {
            Rule r = new Rule(left, rigth);
            this.RuleSet.Add(r);
            this.GenerateAlternatives();
        }

        public string First(Rule r)
        {
            string output = "";
            foreach (var nonTerm in this.NonTerminals)
            {
                if (nonTerm == r.Right[0])
                {
                    foreach (var rule in this.RuleSet)
                    {
                        if (rule.Left == nonTerm)
                        {
                            output += this.First(rule);
                        }
                    }
                }
            }


            foreach (var term in this.Terminals)
```

```
            {
                if (term == r.Right[0])
                {
                    output += r.Right[0].ToString();
                }
            }
            return output;

        }

        public void GenerateAlternatives()
        {
            var alts = new List<Alternative>();

            foreach (char nonTerm in this.NonTerminals)
            {
                foreach (Rule r in this.RuleSet)
                {
                    if (nonTerm == r.Left)
                    {
                        var buffer = "";
                        buffer = First(r);

                        foreach (var c in buffer)
                        {
                            alts.Add(new Alternative(nonTerm, c, r.Right));
                        }
                    }
                }
            }

            foreach (var term in this.Terminals)
            {
                alts.Add(new Alternative(term, term, "0"));
            }

            this.Alternatives = alts;
        }

        public bool Parse(string input, bool silent = true)
        {
            string buffer = "S";
            bool validChar = false;
            bool ruleFound = false;

            do
            {
                foreach (var t in this.Terminals)
                {
                    if (input[0] == t)
                    {
                        validChar = true;
                    }
                }
```

```
                if (!validChar)
                {
                    return false;
                }
                else
                {
                    validChar = !validChar;
                }
                for (int r = 0; r < this.Alternatives.Count; r++)
                {
                    if (this.Alternatives[r].NonTerm == buffer[0] &&
this.Alternatives[r].Term == input[0])
                    {
                        buffer = this.Alternatives[r].Replacement +
buffer.Remove(0, 1);

                        if (!silent)
                        {
                            Console.WriteLine($"Buffer: {buffer}, input:
{input}");
                        }

                        ruleFound = true;

                    }
                }

                if (buffer[0] == '0')
                {
                    input = input.Remove(0, 1);
                    buffer = buffer.Remove(0, 1);
                }
                if (!ruleFound)
                {
                    return false;
                }

                ruleFound = !ruleFound;

            }
            while (input.Length != 0 && buffer.Length != 0);

            if (buffer.Length == 0 && input.Length == 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```