Assignment 5

CAS CS 320: Principles of Programming Languages

Due: **Friday March 8, 2024 by 11:59PM**

Submission Instructions

- You will submit a `.pdf` of your solutions on Gradescope. Your solutions must be legible. If you do not have neat handwriting, please type your solutions.
- **Put a box around the final answer in your solution.** Or otherwise make the final answer in your solution abundantly clear.
- Choose the correct pages corresponding to each problem in Gradescope. Note that Gradescope registers your submission as soon as you submit it, so you don't need to rush to choose corresponding pages. **For multipart questions, please make sure each part is accounted for.**

We will dock points if any of these instructions are not followed.

1 ABCs

Consider the following BNF Grammar.

```
<START> ::= <a>|<b>|<c>
<a>     ::= A<b>|AA<b>
<b>     ::= B<b>|AB<b>|AB<c>
<c>     ::= C|C<c>B
```

A. Give two different derivations of the sentence `AABABC`.

B. Find every (derivable) sentence with 5 terminal symbols. Circle (or otherwise mark) any sentences which can be derived in multiple ways.

Solution.

A

```
<START> => <a> | <b> | <c>
        => A<b>
        => AAB<b>
        => AABAB<c>
        => AABABC
```

```
<START> => <a> | <b> | <c>
        => AA<b>
        => AAB<b>
        => AABAB<c>
        => AABABC
```

B (underline instead of circle)

- `AAABC`
- `ABABC`
- `ABCCB`
- `BBABC`
- `CCCBB`

2 BNF Derivations

Consider the following BNF Grammar.

```
<PROG>  ::= <stmts>
<stmts> ::= <stmt>|<stmt> <stmts>
<stmt>  ::= let <var> = <val>
<var>   ::= x|<var>'
<val>   ::= <var>|fun <var> -> <val>|(<val> <val>)
```

Give a derivation of the sentence

```
let x = fun x -> x let x' = (x x)
```

You may apply rules in parallel, e.g., you may write

```
let <var> = <var>  ==>
let x = x
```

instead of

```
let <var> = <var>  ==>
let x = <var>      ==>
let x = x
```

but you cannot write

```
let x = <val>  ==>
let x = x
```

because this would apply the rule for `<val>` and then the rule for `<var>` on nonterminal symbols appearing in the same position.

Solution.

```
        <PROG> => <stmts>
               => <stmt> <stmts>
               => let <var> = <val> <stmts>
               => let x = fun <var> -> <val> <stmts>
               => let x = fun x -> x <stmt>
               => let x = fun x -> x let <var> = <val>
               => let x = fun x -> x let x' = (<var> <var>)
               => let x = fun x -> x let x' = (x x)
```

3 Integer Literals

Complete the BNF grammar below for integer literals based on the following informal rules.

- A positive integer literal is any sequence of digits `0` through `9`, not starting with the digit `0`.
- A negative integer literal is a positive integer literal with a negation symbol '`-`' in front of it.
- An integer literal is `0`, a positive integer literal, or a negative integer literal.

```
<INT>    ::= ???
<pdigit> ::= 1|2|3|4|5|6|7|8|9
<digit>  ::= 0|<pdigit>
<digits> ::= ???
<pint>   ::= ???
<nint>   ::= ???
```

`<digits>` represents any sequence of digits, `<pint>` represents positive integers, and `<nint>` represents negative integers.

Solution.

```
        <INT>    ::= 0 | <pint> | <nint>
        <pdigit> ::= 1|2|3|4|5|6|7|8|9
        <digit>  ::= 0|<pdigit>
        <digits> ::= <digit> | <digit> <digits>
        <pint>   ::= <pdigit> | <pdigit> <digits>
        <nint>   ::= -<pint>
```