Name:
Yinxuan Feng

BU ID (no dashes):
U17037273

CS 320—Week 8 Homework—Due W 3/27 11:59pm

Write your answers to the problems in the space indicated. Scan your solution and submit to Gradescope as a PDF file. You will receive an email about the Gradescope account. You may do this from your phone using free scanning apps, or with a desktop scanner. Do NOT edit this file and move things around, the format must remain the same.

Problem One (Monad Do Expressions)

This problem will exercise your understanding of the "assembly language" of Haskell's **do** expression syntax. "Translation" in this exercise refers to converting between the forms (i), (ii) and (iii) shown on the last page. Use bound variables x, y, and z (as necessary).

(A) Show what phase (i) of the translation for example `ex9'` in `MonadLectureCode2.hs` would look like (this is in the Maybe Monad).

```
incm 5 >>= (\x -> incm x >>= (\y -> incm y ))
```

(B) Show what phases (i) and (ii) of the translation for example `ex14` in `MonadLectureCode2.hs` would look like (this is in the Maybe Monad).

i)
```
Just (3,4) >>= (\(y1, y2) -> split' x >>= (\(z1,z2) -> return (y1*z1+y2*z2)))
```

ii)
```
Just (3,4) >>= \(y1, y2) ->
split' x   >>= \(z1, z2) ->
return (y1*z1 + y2*z2)
```

(C) Show what phases (i) and (ii) of the translation for example `ex4` in `MonadLectureCode3.hs` would look like (this is in the Checked Monad).

i)
```
divide 1000 2 >>= (\x -> tooLargeWarning x >>= (\z -> divide x 2 >>=
(\y -> sub y 1 )))
```

2)
```
divide 1000 2 >>= €\x ->
tooLargeWarning x >>= \z ->
divide x 2 >>= \y ->
sub y 1
```

```
S -> E
E -> E + T | T
T -> T * F | F
F -> P ^ F | P
P -> - P
P -> 1 | 2 | 3
```

Problem Two (Derivations and Parse Trees)

This problem concerns context-free grammars and the relationship between parse trees and derivations, using the grammar shown at right.

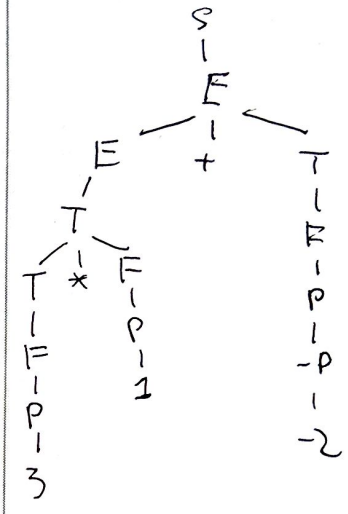
(A) Give a left-most derivation of the string
3 * 2 + - 3 ^ 1 .

S ⇒ E ⇒ E+T ⇒ T+T ⇒ T*F+T
⇒ F*F+T ⇒ P*F+T ⇒ 3*F+T ⇒ 3*P+T
⇒ 3*2+T ⇒ 3*2+F ⇒ 3*2+P^F
⇒ 3*2+-P^F ⇒ 3*2+-3^F ⇒ 3*2+-3^P
⇒ 3*2+-3^1

(B) Give a right-most derivation of the string
3 * 2 + - 3 ^ 1 .

S⇒E ⇒ E+T ⇒ E+F ⇒ E+P^F ⇒ E+P^P ⇒ E+P^1
⇒ E+-P^1 ⇒ E+-3^1 ⇒ T+-3^1 ⇒ T*F+-3^1
⇒ T*P+-3^1 ⇒ T*2+-3^1 ⇒ F*2+-3^1 ⇒ P*2+-3^1
⇒ 3*2+-3^1

(C) Give a parse tree for the string
3 * 1 + - 2 .



(D) Suppose we consider the parse tree you created in part (C). If you walk around the tree in preorder, and each time you touch a non-terminal, you add a derivation step to a derivation, what kind of derivation would result?

left-most derivation

(E) Considering the same process as in (D), what kind of traversal of a tree would correspond to a right-most derivation (see at the link on traversals posted with lecture 2)?

reversed pre order

In (A) and (B) you do not need to give the number of the rule, nor underline the non-terminal being rewritten at each step.

See the YT video for hints on how to do (D) and (E).

(F) Give a short, informal proof of the following statement: If a grammar is not ambiguous, then for any string **w** in the language, every derivation of **w** has the same length (same number of derivation steps). Hint: think about the relationship between derivations and parse trees.

As the grammer is not ambiguous, the parse tree is the same. Different derivations start and traverse the tree in different ways, the the tree is the same. Therefore, all derivation has same length.

Problem Three (Context-Free Grammars and Languages)

This problem will have you write context-free grammars and also think about how to characterize context-free languages.

For parts (A) and (B), give an intuitive description of the language generated by the given context-free grammars, where T = { a, b }.

(A) `S -> a A | b S  A -> a S | b A | ε`

Start from a string that starts with a or b. If starts with a, string can end, or follow an "a" and another char at least. or follow a "b" then it can end. It will end with "b" if length ≠ 1. It will have at least length 2 if it starts with a "b". There is only odd number of a.

(B) `S -> a S b S | b S a S | ε`

~~Starts with a~~. Whenever it adds an "a", it adds an "b", vice versa. So the string has equal number of a and b.

For parts (C) and (D), give a context-free grammar for the language specified.

(C) The language of matching delimiters over the alphabet:

`{ } [ ] ( )`

The following are in the language: `(()) ({}) {()}{}`
and the following are not: `({)} {{{}}} ε`

S → E'
E' → (E') | {E'} | [E'] | [] | {} | () | E'E'

~~E' → (E') | {E'} | ε | [E']~~

(D) The language { $a^n b^m a^n$ | n ≥ 0 and m ≥ 1}, i.e., strings aaa..abbb…baaa..a with at least one b, and starting and ending with substrings of a's of the same length.

The following are in the language: `b  abbbba  aaabbaaa`
and the following are not: `aaba  aaaa  ε`

S → E
E → aE'a
~~E' → aE'a | bE' | b~~

E → b | aE'a
E' → E | bE