Name:

Student ID:

Compiler Construction, Spring 2022
Midterm

1. Please answer the following questions.
   a. What is a compiler? (3 pt)

批改方式:
- 一定要寫到compiler is a program/software that…. , 很多同學只寫功能給2分

A computer program/software that translate computer code written in one programming language (eg: high-level language C++), into a set of machine language instructions.

   b. Which part of a compiler that takes as input a stream of characters and produces as output a stream of words along with their associated syntactic categories? (4 pt)

批改方式:
- 答案除了scanner及lexical analyzer, 其餘都不對

Scanner, a compiler's scanner scans a character-based input stream and creates a word-based output stream, with each word identified with its Syntactic category.

   c. What are the potential data fields in a symbol table? (4 pt)

批改方式:
- 有提到下列2個答案滿分
- 只說明用AST/link list/list/bs等t儲存資料 2分, 需要提到type & identification
- regular expression不對

identification, symbol, type, scope, class

   d. What are the problems that would lead to prediction conflicts in a top-down parser? (4 pt)

批改方式:
- 只提到ambiguity 2分, 有舉出例子/有多加說明滿分
- 列出下列原因, 1個2分

Left recursion 2pt
common prefix 2pt

   e. Please give a brief description of compiler frontend and backend. Which one is machine dependent? In addition, please write down what is used to connect frontend with backend. (5 pt)

批改方式:
- frontend backend只寫machine independent給1分, 需提供解釋包含什麼細項
- backend只說明他是machine dependent不完整, 需提供解釋包含什麼細項
- 連接frontend & backend只接受ABS Tree的答案, 其餘不給分(請參考ppt)
- 這題有4小題, 漏寫不給分

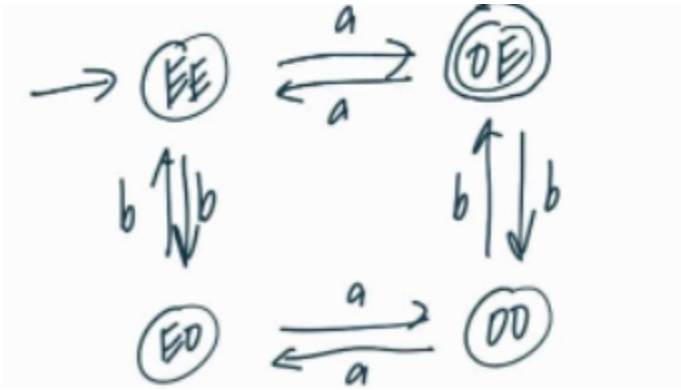
Frontend: 主要執行syntax analyze (ex: scanner parser) 1.5pt
Backend: 主要執行機器碼合成部分 (ex: code generation) 1.5pt
Backend phase 1pt
abstract syntax tree 1pt

2. Please give your answer to each of the following questions. You do not need to show how to derive the answer.
   a. For the following language, please construct a Deterministic Finite Automata (DFA). (10pt)
   A = {w ∈ {a, b} * : odd number of a's , and even number b's}.

批改方式:

- 批改會檢查是否符合 {a, bb, abb, aaabb, bba, bbaaa…}
- 沒有清楚表明start箭頭扣2分
- 畫錯0分

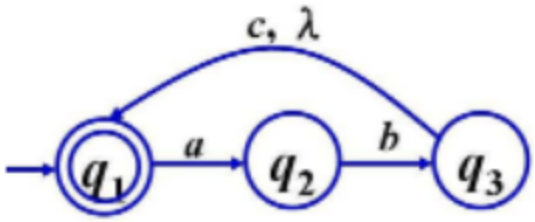
ANS



b. Construct a Non-Deterministic Finite Automata (NFA) with three states that accept the language:
{ab, abc}*. (5pt)

批改方式:

- 批改會檢查是否符合 {lambda, ab, abc, ababc, abcab…}
- 畫錯0分
- 沒有start箭頭扣2分
- 注:first state是final



3. Write a Context-Free Grammar for the language composed of all binary numbers, containing at least three consecutive 1's. For example, the language will have the strings: 01110010, 11101111111, 111, but not 0000110101. (10pt)

ANS
S -> A 111 A
A -> 0A | 1A | λ

Context-Free Grammar-
(0|1)*111(0|1)*

4. Consider the following lex-like specification. There are six token classes. The alphabet is the set {a, b, c}. Parentheses are used to show the association of operations and are not part of the input alphabet.

abc	{return TokenClass 1;}
(b | c)*	{return TokenClass 2;}
a*	{return TokenClass 3;}
(abc)*	{return TokenClass 4;}
aab*	{return TokenClass 5;}
(b | c)c	{return TokenClass 6;}

   a. Show how the following string would be partitioned (by adding vertical lines) into tokens by grouping the characters into lexemes. Label each lexeme with the integer of the correct token class. For example, the answer will look like the sequence: 135461. (10 pt)
   baaabccacccabbaaaabcccbccaaacccabcabcabc

   b. Can any of these rules be removed without changing the scanner's behavior on any string? Please justify your answer. (5 pt)

ANS

(a) b | aaa | bcc | a | ccc | a | bb | aaaa | bcccbcc | aaa | ccc | abcabcabc

2 | 3 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 4

(b) (b | c)c can be removed, since (b | c)* can be used instead.
abc can be removed, since (abc)* can be used instead.

評分方式："2 取代 6" 和 "4 取代 1" 這種"一對一"的替換，至少寫到其中一個且觀念正確就至少4分。若用 2 & 3 來取代其它如 4, 5, 可能會改變scanner's behavior, 所以多寫會扣1。如果有自己定義優先權，且描述合理，沒寫一對一的替換不扣分。

5. Please design a ac (adding calculator) language.
   a. Filling out (A) to (G) in Table 1 uses regular expressions to define the ***ac*** tokens, i.e., assign, plus, minus, fnum (floating point numbers), and id (identifiers, which could be any alphabetic character exc31o;ept the reserved character for *floatdcl*, *intdcl* and *print*). (5pt)
   b. Given input, draw a leftmost derivation parse tree. (A derivation always chooses the leftmost possible nonterminal at each step is called a leftmost derivation)(10pt)

Input: f x i y x=9 y =x+5 x=x/3.33*y p x p y p x

Terminal	Regular Expression
floatdcl	"f"
intdcl	"i"
print	"p"
id	(A)
assign	(B)
plus	"+"
minus	-"
inum	(C)
fnum	(D)
blank	(E)

```
 1| Prog  -> Dcls Stmts $
 2| Dcls  -> Dcl Dcls
 3|        | λ
 4| Dcl   -> floatdcl id
 5|        | intdcl id
 6| Stmts -> Stmt Stmts
 7|        | λ
 8| Stmt  -> id assign Val Expr
 9|        | print id
10| Expr  -> Term
11|        | plus Val Expr
12|        | minus Val Expr
13| Term  -> Val
14|        | multiply Val Expr
15|        | div Val Expr
16|        | λ
17| Val   -> id
18|        | inum
19|        | fnum
```

data field

ANS

(a) A – E are following respectively

id $[a-e] \mid [g-h] \mid [j-o] \mid [q-z]$

assign "="

inum $[0-9]^+$

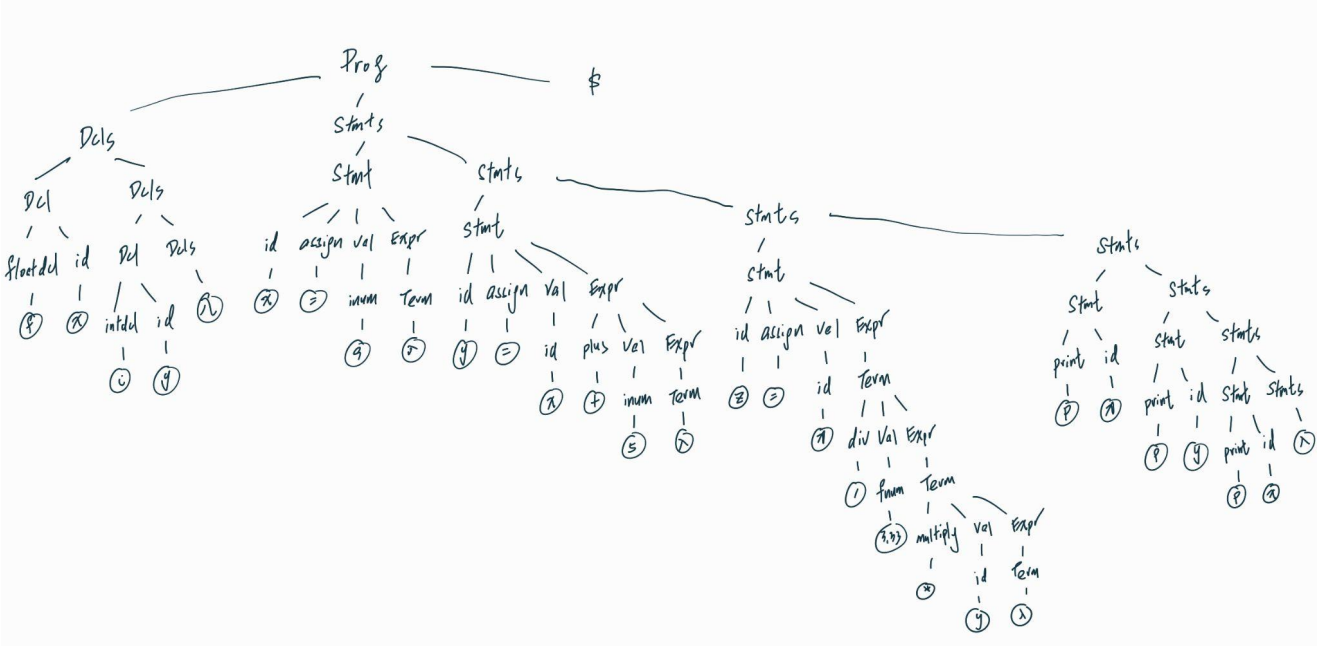
fnum $[0-9]^+.[0-9]^+$

blank $(" ")^+$

(b)

批改方式：

- 少寫一個step扣1分



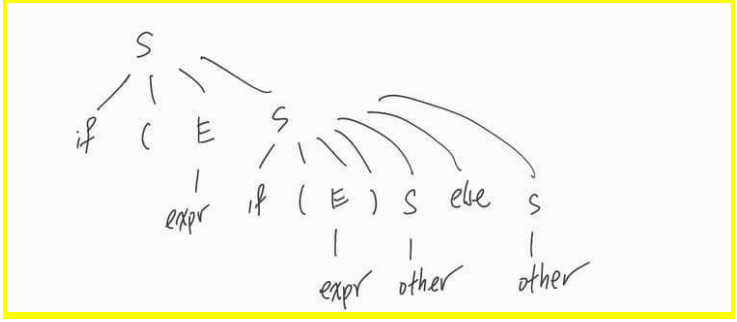
6. Consider the grammar G, and the input string.
   a. Give the rightmost derivation for the string. (4 pt)
   b. Dive the leftmost derivation for the string. (4 pt)
   c. Is it a ambiguous grammar? why? (2 pt)

Grammar G:

S -> if (E) S | if (E) S else S
S -> other
E -> expr

Input: if (expr) if (expr) other else other

ANS



(a) Leftmost-

```
S -> if ( E ) S
  -> if ( expr ) if ( E ) S else S
  -> if ( expr ) if ( expr ) other else other
```

(b) Rightmost-

```
S -> if ( E ) S else S
  -> if ( E ) S else other
  -> if ( E ) if ( E ) S else other
  -> if ( E ) if ( E ) other else other
  -> if ( E ) if ( expr ) other else other
  -> if ( expr ) if ( expr ) other else other
```

(a) Yes, the leftmost and rightmost tree are having the same output.

7. Building an LL(1) parser needs to check the ambiguity of the given grammar. Later, compute First, Follow, and Predict sets in order to build the table-driven LL(1). Please answer the following questions based on the examples below,and add the symbols λ and $ if necessary.
   a. Given the following context free grammar, compute First, Follow, and Predict sets. (15pt)
   b. Build the LL(1) parse table with information you have in (a). (5 pt)
   c. Show parsing process for the given input string. (15 pt)
   (**Note:** Please write down your answers in the following formats.)

   input : d a h g $

   Context-free grammar:

```
1| S -> A C B $
2| A -> d a
3|    | B C
4| B -> g
5|    | λ
6| C -> h
```

Example format for First, Follow, and Predict sets.

0|	Grammar	First()	Follow()	Predict set
1|	S -> A B C $	?	?	?
2|	A -> a D	?	?	?
3|	D -> C	?	?	?

Example format for your parse table.

Non-terminals	**a**	**b**	**c**
S	…	…	…
A	…	…	…
…	…	…	…

Example format for the parsing procedure.

Parse Stack	**Action**	**Remaining Input**
S	Apply1: S -> ABC$	abc$
$CBA	Apply2: A -> a	abc$
...	…	…

a.

錯一格扣1分，同一個Non-Terminal的Follow扣一次就好

Rules	First	Follow	Predict
1| S -> A C B $	d g h	$	d g h
2| A -> d a	d	h	d
3| | B C	g h		g h
4| B -> g	g	$ h	g
5| | λ	λ		$ h
6| C -> h	h	$ g h	h

b.

Non Terminal	Terminal				
	a	d	g	h	$
S		1	1	1	
A		2	3	3	
B			4	5	5
C				6	

c.

Action跟Remaining Input的順序可整欄上下平移，因為題目沒規定同一列是做完Action後得到Remaining Input，還是看到Remaining Input後再去做該Action。

下表參考答案是看到Remaining Input後再去做Action，並把做完該Action後的Remaining Input放到下一列。

另外注意Parse Stack, 如果同學有額外定義Top of Stack的位置, 就依照同學定義。
若沒有就照投影片。

Parse Stack (Bottom of Stack→Top of Stack)	Action	Remaining Input
S	Apply `1| S -> A C B $`	d a h g $
$ B C A	Apply `2| A -> d a`	d a h g $
$ B C a d	Match	d a h g $
$ B C a	Match	a h g $
$ B C	Apply `6| C -> h`	h g $
$ B h	Match	h g $
$ B	Apply `4| B -> g`	g $
$ g	Match	g $
$	Accept	$