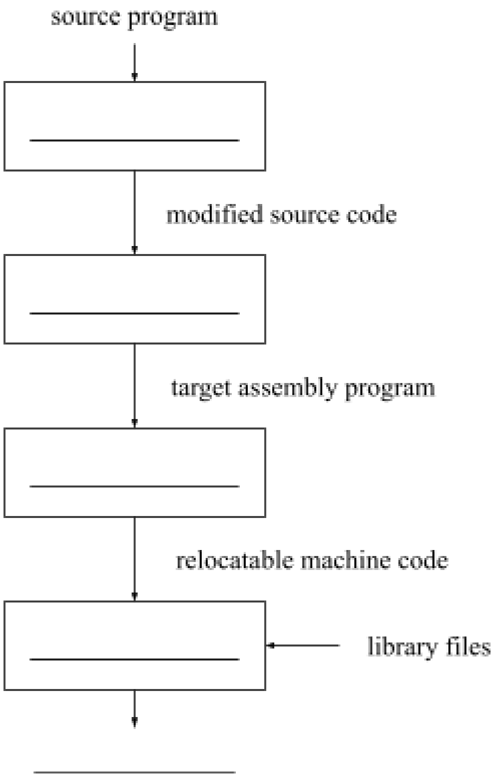
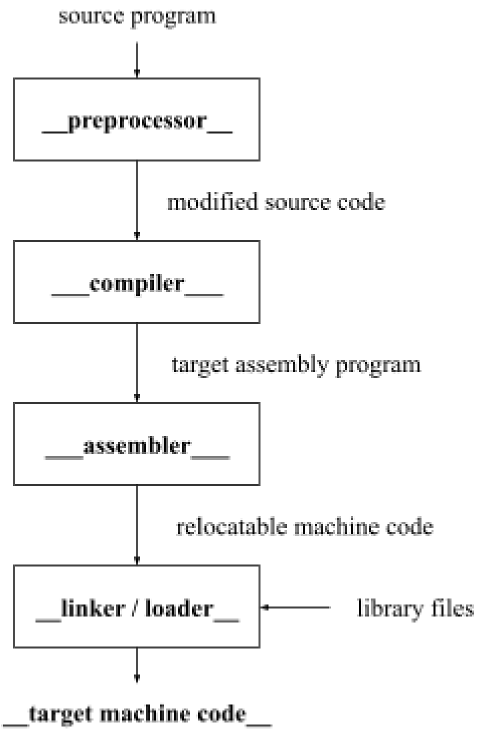
<u>2024 NCKU CSIE Compiler Midterm Exam</u> **(total score: 120)**

Student ID: ____________ Name: ____________

1. (10%) Compiler Process. Please fill in the blanks below. (2% * 5 blanks)



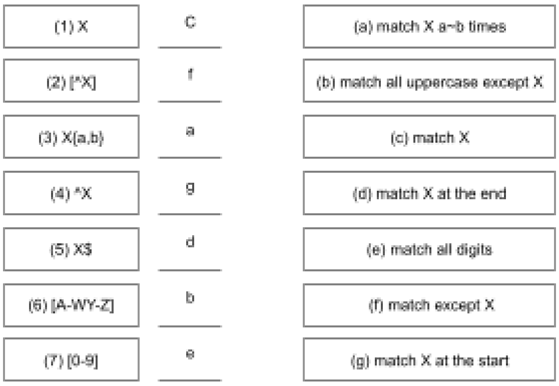


ans:

- 沒有特別標示順序則由上往下對照批改

2. (25%) Regular Expression.
<u>limitation:</u>
i). all your answer Regular Expression should < **36** characters
ii). enumerating all cases and predefined variable are **not allowed**

a. (5%) choose corresponding (a) ~ (g) for sub-question (1) ~ (7) (hint!)
ans: c f a g d b e



(1) X	c	(a) match X a~b times
(2) [^X]	f	(b) match all uppercase except X
(3) X{a,b}	a	(c) match X
(4) ^X	g	(d) match X at the end
(5) X$	d	(e) match all digits
(6) [A-WY-Z]	b	(f) match except X
(7) [0-9]	e	(g) match X at the start

b. (5%) All strings that only contain lowercase a **and** b, except string "ab"
   - For example: "ba", "aaaaaaaab" , "abababaa", "bbabab"
   - Not including, for example: "a", "b", "ab", "abc", "aaaaaa"

   ans: a[ab]+b | [ab]*ba[ab]*
   ans: [ab]*(abb | ba | aab)[ab]*
   ps: [ab] = (a|b)

c. (5%) Defines a C-like, fixed-decimal literal with no superfluous leading or trailing zeros.
   - For example: 123.456, 1200.08, 1.0, 0.0
   - Not including, for example: 114514, 000.00, 120.800, 001.100

   ans: (0 | [1-9][0-9]*)\.(0 | [0-9]*[1-9])

d. (5%) Please represent with a Regular Expression any string that meets all of the following criteria: **Starts** with an 'A', followed by **three to five** digits, **then** a hyphen '-', and **ends** with **four** uppercase English letters.
   - For example: "A23651-TSMC", "A231-APEX", "A8093-PTSD"
   - Not including, for example: "aA23651-ADCD", "A19-COVID", "A23651-GuRa", "P7612-compiler"

   ans: ^A[0-9]{3,5}-[A-Z]{4}$

e. (5%) Please represent with a Regular Expression any string that meets all of the following criteria: The string with substring that **one or more** uppercase English letters, followed immediately by **exactly one** non-digit.
   - For example:
     in "regADDSC#$$@", the substring "ADDSC#";
     in "face_me@SEKIRO!", the substring "SEKIRO!";
     in "a!~DF$ac", the substring "DF$";
     in "bb()ZELDa", the substring "ZELDa"

   ans: [A-Z]+[^0-9]

3. (25%) DFA / NFA. (for b. and c., you only need to present the automata graph)

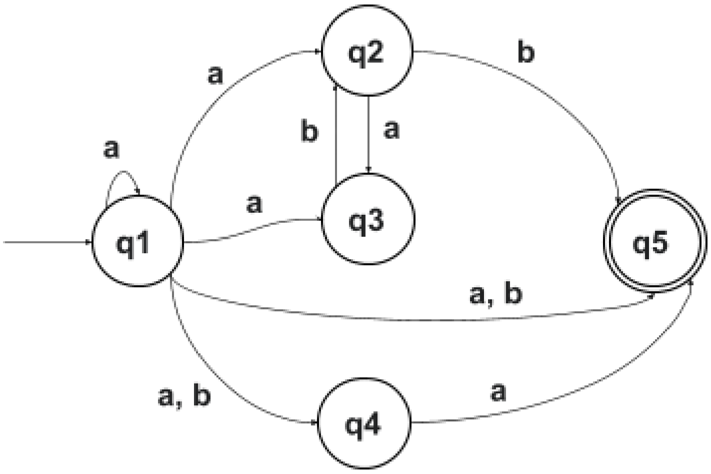
a. (5%) Please explain the difference between DFA and NFA in two aspects:

- as for Determinism vs. Nondeterminism. (3%)
  ans: 提到每一個 **token** 都只有一種可能的 **transition** 即給分

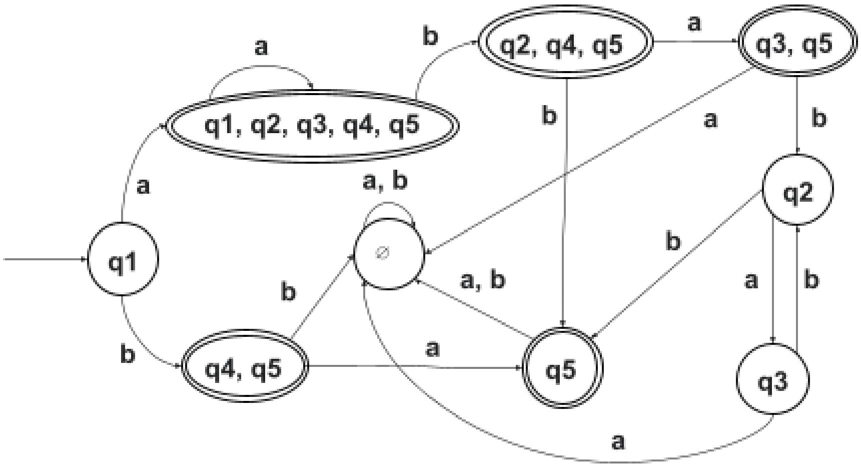
- as for performance considerations (time / the number of states / construction…). (2%)
  ans:
  就時間而言, 由於 **DFA** 的每個 **state** 對每個輸入 **token** 的 **transition** 都是預先定義且只有一種可能, 無需在運行時進行選擇或回溯因此所耗時間通常比 **NFA** 少。

  就 **state** 的數量和建構複雜度而言, 若沒有特定前提就沒有標準答案, 會依據你提出原因的合理性批改。
  1. 因為 **NFA** 有非確定性的性質(即不只一種可的 **transition**）和 **ε-transition** , 讓 **transition** 可以無限擴增, 使得 **state** 數量通常比 **DFA** 多。但也因以上性質, 對人類來說較易讀, 也不需將所有 **transition** 畫出來, 建構較 **DFA** 簡單。
  2. 如果以任意 **DFA** 都是 **NFA** 來看, 那麼 **NFA** 的 **state** 數量會小於等於 **DFA** 的 **state** 數量, 一個 **NFA** 可以被轉換成一個等價的 **DFA**, 即兩者識別相同的 **RE**。在轉換過程中, 對原有 **n** 個狀態的 **NFA** 來說, 其等價的 **DFA** 最多可能有 2^n 個狀態。轉換後, 對電腦來說較易判讀, 速度也較快。

b. (10%) Transform below NFA to the **minimized** DFA.



ans:

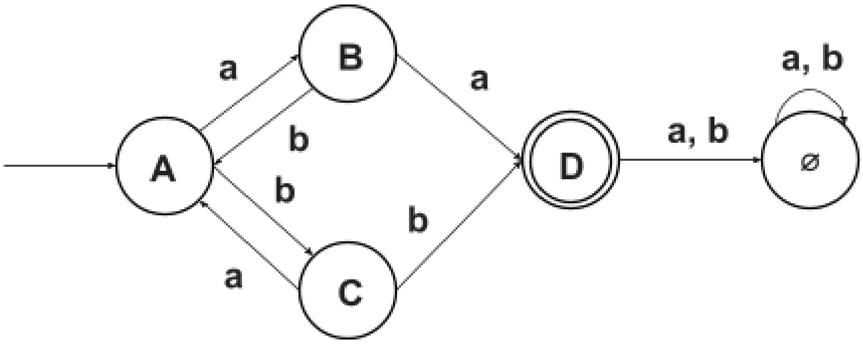


- 空集合 state 沒畫到, 其他 state, edge 皆正確扣 3%
- 沒有畫到 input edge、accept 的雙圈圈皆扣 2%
- 少 edge / 多 edge / 簡化錯 state(新 state 的集合有多 / 少舊 state)皆扣 2%
- 空集合 state 沒有回到自己的 edge(吃 a, b)扣 1%
- state 數、edge 數對, 定義集合寫錯扣 5%

c. (10%) Transform below Regular Expression to the **minimized** DFA.

$(ab|ba)^*(aa|bb)$

Hint: you'll finally have **less than 10 states** when the DFA is minimized, and you should clearly write down your definition of every state in the automata.



- 由於提示的 state 數量有誤, 小於 10 個但非最簡酌情扣 5-7%
- 空集合 state 沒畫到, 其他 state, edge 皆正確扣 3%
- 空集合 state 沒有回到自己的 edge（吃 a, b）扣 1%
- 沒有畫到 input edge、accept 的雙圈圈皆扣 2%
- aa, bb 不能 accept 則全扣

4. (10%) Let G be the grammar:

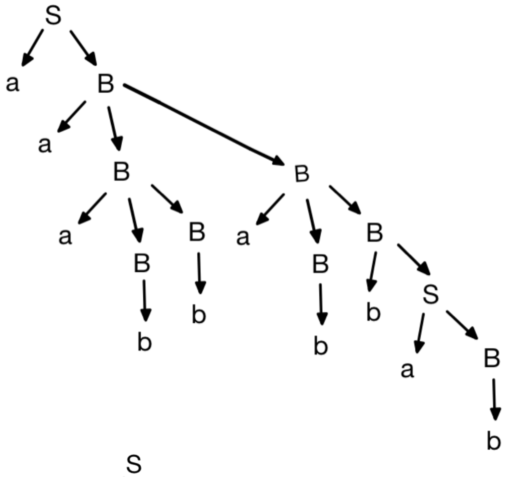
S → aB|bA
A → a|aS|bAA
B → b|bS|aBB

Find the leftmost derivation (3% *2) and parse tree (2% *2) for each following string.

a. (5%) aaabbabbab

ans:

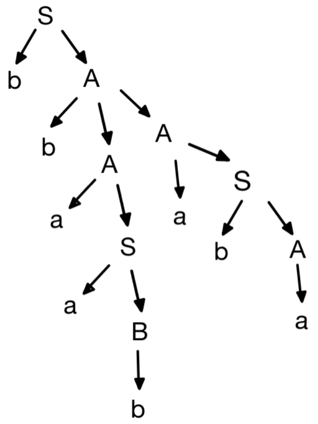
S → aB
→ aaBB
→ aaaBBB
→ aaabBB
→ aaabbB
→ aaabbaBB
→ aaabbabB
→ aaabbabbS
→ aaabbabbaB
→ aaabbabbab



b. (5%) bbaababa

ans:

S → bA
→ bbAA
→ bbaSA
→ bbaaBA
→ bbaabA
→ bbaabaS
→ bbaababA
→ bbaababa

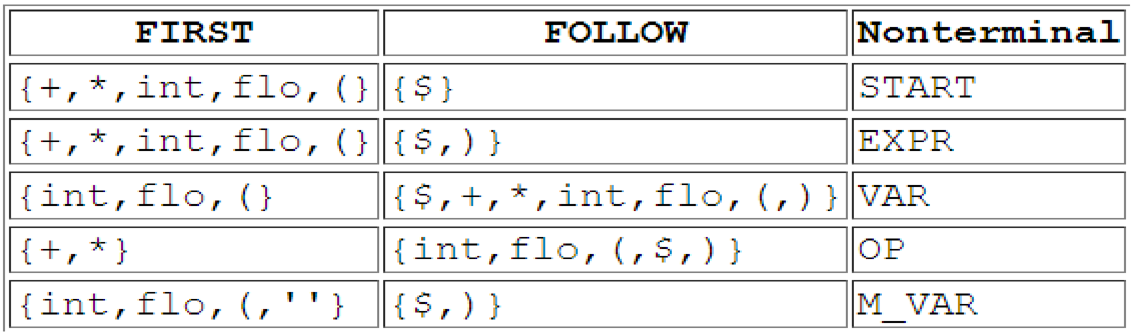


- 多寫出其他 parse 的分支扣 1%
- 直接拆解 A → a|aaB|abA|bAA… 扣 2%
- 沒有遵守 leftmost 步驟(跳太快)扣 1%

5. (40%) For the following grammar, please do top-down parsing:

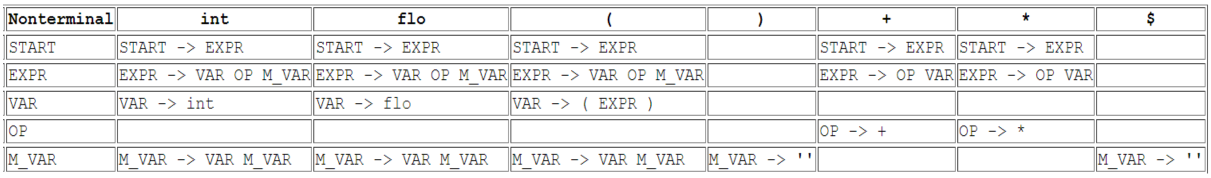
START → EXPR $ START → (1)
EXPR → OP VAR | VAR OP M_VAR EXPR → (2) | (3)
VAR → int | flo | (EXPR) VAR → (4) | (5) | (6)
OP → + | * OP → (7) | (8)
M_VAR → VAR M_VAR | λ M_VAR → (9) | (10)

a. (15%) First set
b. (15%) Follow set
c. (10%) LL(1) parsing table

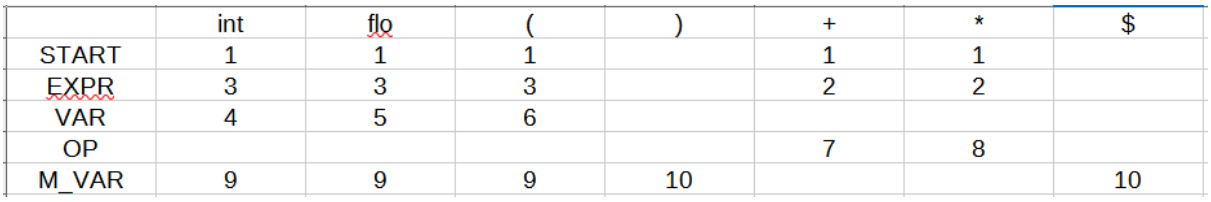


FIRST	FOLLOW	Nonterminal
{+,*,int,flo,(}	{$}	START
{+,*,int,flo,(}	{$,)}	EXPR
{int,flo,(}	{$,+,*,int,flo,(,)}	VAR
{+,*}	{int,flo,(,$,)}	OP
{int,flo,(,''}	{$,)}	M_VAR

註:答案中 FOLLOW(START) 要改為{ λ }



Nonterminal	int	flo	(	)	+	*	$
START	START -> EXPR	START -> EXPR	START -> EXPR		START -> EXPR	START -> EXPR	
EXPR	EXPR -> VAR OP M_VAR	EXPR -> VAR OP M_VAR	EXPR -> VAR OP M_VAR		EXPR -> OP VAR	EXPR -> OP VAR	
VAR	VAR -> int	VAR -> flo	VAR -> (EXPR)				
OP					OP -> +	OP -> *	
M_VAR	M_VAR -> VAR M_VAR	M_VAR -> VAR M_VAR	M_VAR -> VAR M_VAR	M_VAR -> ''			M_VAR -> ''



	int	flo	(	)	+	*	$
START	1	1	1		1	1	
EXPR	3	3	3		2	2	
VAR	4	5	6				
OP					7	8	
M_VAR	9	9	9	10			10

6. (10%) Given the grammar (upper cases are non-terminal, and the lower cases are terminal):

A → Au | Ag | Ar | Be | Cs
B → Bo | Ba | Al | Co
C → Cu | Cr | At

Please convert left recursion to right recursion.

A → B e A'
| C s A'
B → C s A' l B'
| C o B'
A' → u A'
| g A'
| r A'
| ϵ
B' → o B'
| a B'
| e A' l B'
| ϵ
C → u C
| r C
| s A' l B' e A' t C
| o B' e A' t C
| s A' t C
| ϵ