Solver: GOH KA HIAN

Email Address: kgoh022@e.ntu.edu.sg

1. a)
Machine code is a set of instructions that is executed directly by a machine CPU.

Assembly Language is a low-level programming language, generally with a one-to-one correspondence with machine code of a processor family.

High level programming language is a human readable language of high abstraction that is usually compiled before being executed by the machine.

b)
(i) False, both DFA and NFA can have finite number of states
(ii) False, both DFA and NFA can definite infinite number of strings
(iii) False, NFA can have state transition without reading any input symbol

c)
i)
Pair 1 is different:
a|bab* cannot define ab while ab* |bab* can define ab

Pair 2 is the same.
b? = b|λ
a(ba)* = (ab)*a

ii)
([a-zA-Z] | '-')* [0-1][0-9]'-'[0-9][0-9][0-9]([a-zA-Z]|'-')*

2. a)

matrix = LBRACKET empty_vectorlist RBRACKET

empty_vectorlist = λ
| vectorlist

vectorlist = vector
| vector SEMICOLON vectorlist

vector = INTLITERALS
| INTLITERALS COMMA vector

b)

predict(p1) = {a,b,c,d,$}

predict(p2) = {a}

predict(p3) = {b,c,d}

predict(p4) = {$}

predict(p5) = {b}

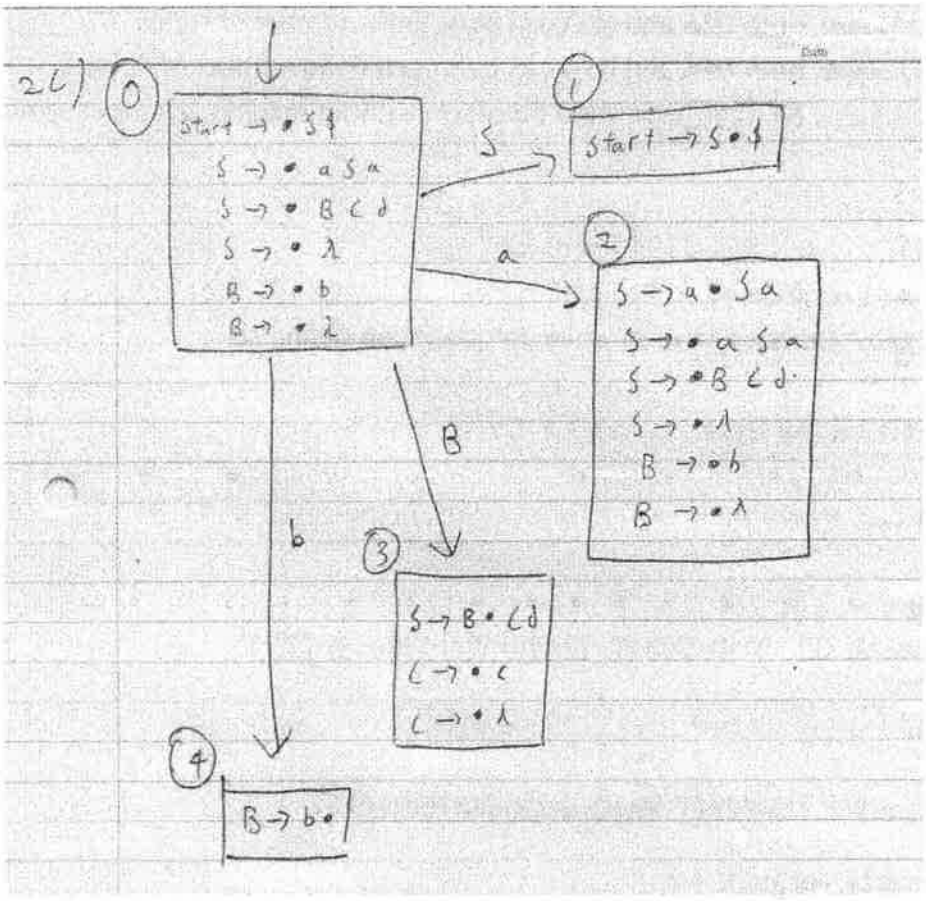
predict(p6) = {c,d}

predict(p7) = {c}

predict(p8) = {d}

Yes, it is in LL(1). (because for a particular LHS, we do not have a similar token appear in two different rule)

c)



Reduce-reduce on λ for both state 0 and 2

3a

i)

```
Frame = Box.b {: return new Frame(b); :}
Box = HORIZONTAL Box.l Box.r {: return new HBox(l,r); :}
| VERTICAL Box.t Box.b {: return new VBox(t,b); :}
| ATOMIC NUM.i NUM.j {: return new ABox(Integer.parseInt(i),Integer.parseInt(j)); :}
```

ii)

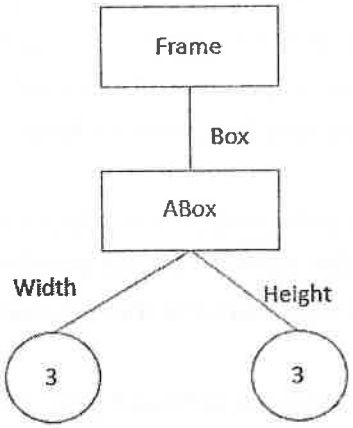
syn int Box.area();

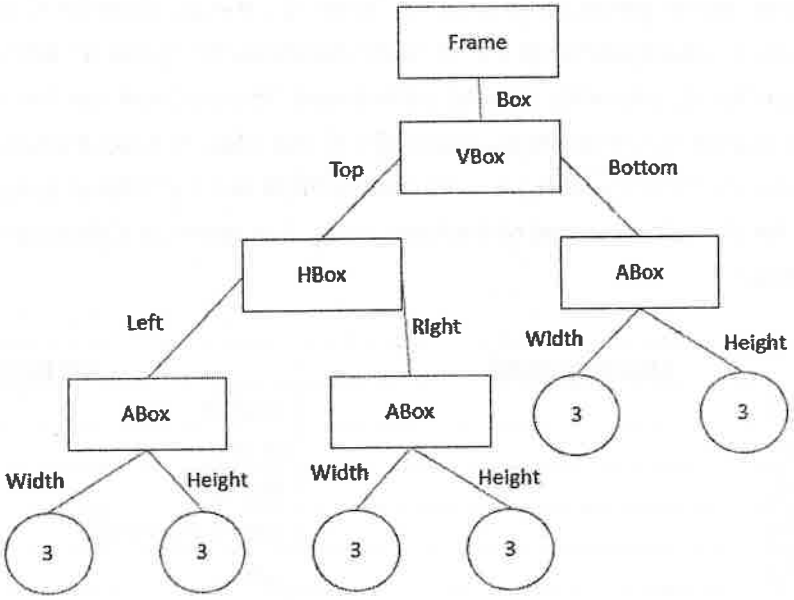
eq VBox.area() = Math.max(getTop().width(), getBottom().width()) * (getTop().height() + getBottom().height());

eq HBox.area() = Math.max(getLeft().height(), getRight().height()) * (getLeft().width() + getRight().width());

eq ABox.area() = getWidth() * getHeight();

iii)





17

b)

(Just give some examples)

Name Checking:

For variable declaration check that there isn't a variable with same name within scope.

Check when using a variable, its name is declared within scope.

Type Checking:

Check IfStmt, WhileStmt - expr is of Boolean type.

Check for assignment statement - type of LHS and RHS matches.

Check for return statement, type of expr matches function return type.

Check parameters of arguments matches a function parameters.

4a)

JIT allows caching of heavily used part of program, improving performance.

Modern JIT compilers also exploit runtime profiling for optimization, hence they can perform more targeted optimisations than compilers targeting native code directly

b)

iload_0 is preferred as it is more compact (less code space)

c i)

Problem: When generating code for "goto l1", the location for l1 is not yet known.

Solution 1 – Backpatching: Put in null references for "goto l1" and once we know the location for l1, overwrite the null references. This method can be costly as we must keep track of null references, especially in the case of nested loop.

Solution 2 – NOP Padding: Generate two NOP instructions as jump targets and insert them for l1. As compared to backpatching, this method is thus better as no tracking is needed.

c ii)

Unoptimized	Optimized
load 0	load 0
load x	load x
Ifgt l1	Ifgt l1
load y	load x (shifted)
load y	load y
store z	load y
load y	~~store z~~
load 1	load y
add	load 1
store y	add
load x	store y
load z	~~load z~~
Sub	sub
store x	store x
load 1	load 1

load 0	load 0
ifgt l0	ifgt l0
load x	load x
load 1	load 1
Add	Add
store y	store y

d)

Perform Chaitin algorithm: try with 3 registers first (push node with less than 3 neighbors)

Stack: 5, 6, 7

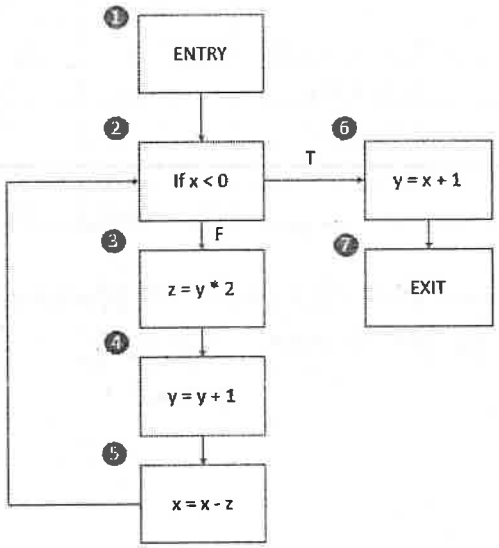
No other nodes can be pushed – spilling required, thus 3 not the answer.

Try with 4 registers (push node with less than 4 neighbors)

Stack: 5, 6, 7, 3, 1, 2, 4

Hence, 4 is the correct answer.

5 a)



b)

x and y is live (exist a path that read them i.e for x: y = x + 1; for y: z = y *2).

z is dead (z is not read in any path)

c)

For available expression

Set of expression = {y * 2, y + 1, x – z, x + 1}

OUT	IN
out(1) = in(1)	in(1) = ∅
out(2) = in(2)	in(2) = out(1) ∩ out(5)
out(3) = in(3) \ {x–z} U {y*2}	in(3) = out(2)
out(4) = in(4) \ {y+1, y*2}	in(4) = out(3)
out(5) = in(5) \ {x–z, x+1}	in(5) = out(4)
out(6) = in(6) \ {y+1, y*2} U {x+1}	in(6) = out(2)

out(7) = in(7)	in(7) = out(6)

d)

Sub IN to OUT

OUT
out(1) = ∅
out(2) = out(1) ∩ out(5)
out(3) = out(2) \ {x–z} U {y*2}
out(4) = out(3) \ {y+1, y*2}
out(5) = out(4) \ {x-z, x+1}
out(6) = out(2) \ {y+1, y*2} U {x+1}
out(7) = out(6)

Solve Iteratively (without worklist algorithm)

out(n) = U = {y * 2, y + 1, x – z, x + 1}; out(1) = ∅;

	0	1	2	3	4
out(1)	∅	∅	∅	∅	∅
out(2)	U	∅	∅	∅	∅
out(3)	U	{y*2, y+1, x+1}	{y*2}	{y*2}	{y*2}
out(4)	U	{x-z, x+1}	{x+1}	∅	∅
out(5)	U	{y*2, y+1}	∅	∅	∅
out(6)	U	{x-z, x+1}	{x+1}	{x+1}	{x+1}
out(7)	U	U	{x-z, x+1}	{x+1}	{x+1}

For reporting of errors and errata, please visit pypdiscuss.appspot.com

Thank you and all the best for your exams! ☺