Q1 Language Standards

1. It is true that the java packages are stored in the same manner as the file system however it is not mandatory to do so.

Why is this done?

- This is very convenient in order to store the packages in a convenient manner
- Secondly, these are the basic industrial standards that if are changed slightly might make it very difficult to interface
- In Implementation, the SE platform uses a hierarchial file system in which each sub-directory would represent the top level of this directory.
- Apart from storage, the entire naming convention of these packages are based on the listing that are used to store the sub-directories. There are some universal standards that are followed. (Just to avoid naming Conflicts).

(b) Java packages can be stored in a file system or any other form of a database
→ Local file system → these have to follow strict constraints.
→ distributed file system → depending upon the implementation.
→ Any other form of database depending upon the use.

2. Two reference types are the same compile-time type if they are declared in the compilation units associated with the same module, and if they have the same binary name.
Even if the type arguments are same, they should have same compile-time type.
a. They are both Class/Interface are defined by the same class loader and have the same binary name in which they are referred as same run time class.
b) Both arrays & their components are in the same run time types.

3. There are basically 2 types of Arguments in the Parameterized classes
   1. Referenced types
   2. Wildcards. (Bounded & Unbounded)

   If a type parameter is a generic one then it is a better idea to use using a wildcard.

4. There are three types of variables that are implicitly Final.

   a) a field of an interface
   b) a local variable declared as a resource of a try-with-resource statement
   c) an exception parameter of a multi catch claus.

5. `strictfp` is the keyword that is used by java in order to avoid floating point errors from platfort to platfom.

example.

```
public strictfp class Tree {

    public static final double PI = 3.14;

    strictfp double Calculation () {
        double R = 10 e+2;
        double R0 = 2 e+4;
        return R + R0;
    }

}
```

6. String objects are intended to be immutable in Java.
To add to this the Java PL depends upon the strings to be immutable in Java. This is because of Java takes strings as literals.

There are 2 ways to overcome this condition in Java
1) String Buffer ✓
2) String Builder x (Not Mentioned)
3. Section 15.18.1, Pg 609. → Java language.

String Buffer → This is used in order to check & increment the string on the fly.
→ This is also used in order to reduce the string buffers as the same strings will point to the same location.

7. Section 8.3.1 (Hidding of Classes & instance variables)

a) ~~Hidding of Class variables~~
To answer the question, yes the classes and variables can be hidden in Java

a) Hidding of Class variable - 8.3.1.1-2.
This is done so that the class x does not inherit the field from its superclass Point. The code in class Test may refer to the same field super.x but the class variable will be hidden.

b) Hidding of Instance Variables - 8.3.1.1-3
In this case the declaration of ~~class~~ x in class test hides the declaration of x in Class point.
The simple name x always refers to the field declared within class test.

8. The point 2. in Section 6.3.10 answers the question
- Yes, the class name / enum name can be hidden by the name of a variable, data member etc in the same scope. If a class / enum na and a variable, function etc are declared in the same scope, the class or enumerator is hidden wherever the variable, funct etc name is visible.

=) A name can therefore be hidden by a explicit declaration of that same name in a nested region or a derived class.

9. In C++, Friendship is not inherited

Lets say that there is a base class with a friend function. This does not mean the function friend has become a function of the derived class

Fore excmp.

```
class. My Assignment
{
friend class. Myotherassignment;
private:
    int val = 100;
};
class .Myotherassignment{
public:
    void show.Val. ( My.Assignment ma)
    {
        increment & fri
    }
};
```

10 Basically the question is, Scope Vs. visibility in C.

C++ → This is an object oriented language. This means that the identifier visibility just mean the parts of the program to which a name, function, variable etc is visible.
Section 5.1 (Pg. 28)

C → In the case of C the visibility of an identifier could mean many thing. ~~Two defens~~
- One definition is the visibility of a identifier within the same scope since the objects are not oriented
- Secondly in the main class, we can only reache once the compile has gone through #include_. This mean that scope is beneath it and can only come into picture once the super classes are run.

11. Public Inheritance
Class A: Public Member, Protected Methods
Public B: Public members, Protected methods (Base Class)

~~Protected Inheritance~~
Class A : Protected Class B
Class B : Public members, Protected Methodes
= Class A : Protected members, Protected Methods

This simply means that, in case of Protected inheritance all the members of the base class become the Protected members of the subclass no matter how they are defined in the base Class.

12.

For loop	For each loop
- We need to specify the bounds to this loop.	- There is no need to specify the bounds in this loop.
- This will keep on executing the block of statements until returned false.	- This will treat everything as a collection & repeats action for every element in an object.
- Generally there is not a lot of overhead.	- There is a larger overhead which decreases the program throughput.

Ques 2. Grammar & Parse tree.

a) $S \rightarrow SS- \mid SS+ \mid a \mid b$

This is a language whose strings all consist either atleast a strings a or b (terminals), or a group of a^n or b^n followed by each other

(b) $S \rightarrow aSaa \mid B$

$B \rightarrow bB \mid \epsilon$

This is a language whose strings all consists of either a^3 ~~or~~ b alone but will allways end with an ~~Empty~~ string (ϵ)

c). $S \rightarrow Aa \mid MS \mid SMA$

$A \rightarrow Aa \mid \epsilon$

$M \rightarrow \epsilon \mid mm \mid bMa \mid aMb$.

This is a language whose strings all consists of atleast an 'a' followed by group of 'a' & 'b' (left recursive) so that the language always ends in terminal a, b and ϵ always. (Empty string)

(d) $S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$.

This is a language whose strings all consists of either 'a' ~~or~~ (a^3) or ~~(b^3)~~ ~~or only a string of a~~ combination of 'a' ~~and~~ 'b' which is optionally followed by an empty string (ϵ).

(e) $S \rightarrow \epsilon \mid SS \mid [S]$

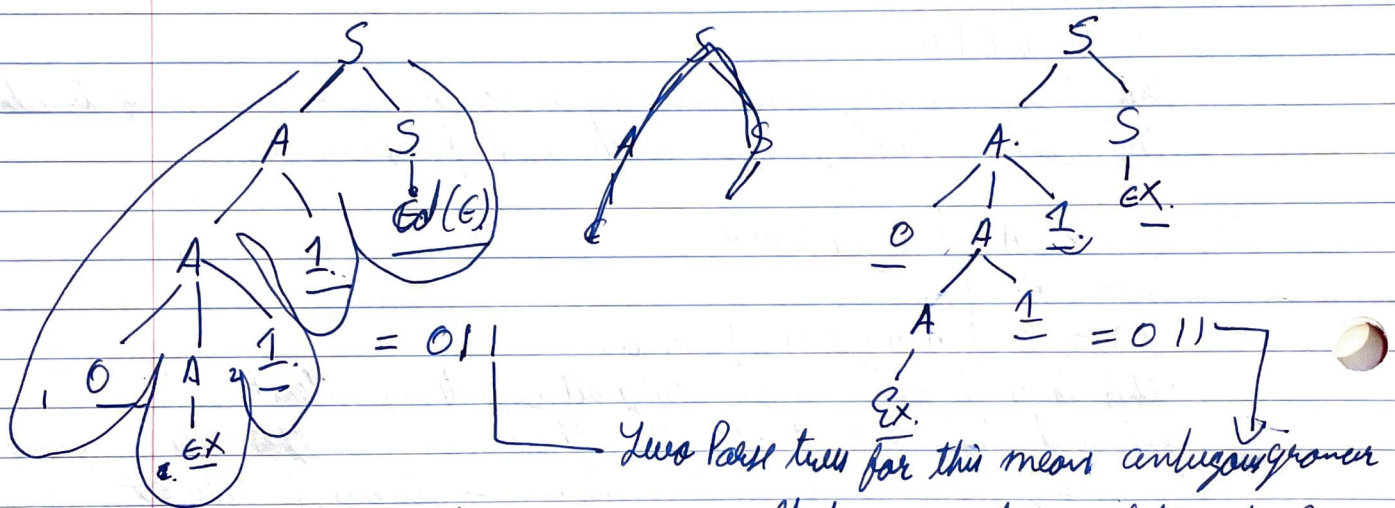
~~This is a language which is followed by~~ contains 's' ~~or group of 's' which will always end in~~ ϵ (~~empty value~~).

This is a language which will allway be a non terminals string as there is no terminals it can recruit to.

2. Grammar.
$S \to A\,S \mid \epsilon$
$A \to A\,1 \mid 0\,A\,1 \mid \epsilon$

(a) Demonstration of a ambiguity in the grammar.

Tree 1.



Two Parse trees for this mean ambiguous grammar

→ In this example we can see that we got a different Parse tree because Associativity and Precedence were not taken care of.

→

(b) We can overcome the above condition by changing the grammar a bit and making it more specific

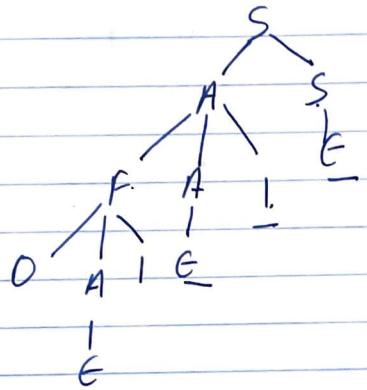
$S \to A\,S \mid \epsilon$
$A \to A\,1 \mid 0\,A\,1 \mid \epsilon$

$S \to A\,S \mid \epsilon$
$A \to F\,A\,1 \mid \epsilon$
$F \to 0\,A\,1 \mid \epsilon$

~~$S \to A\,S \mid \epsilon$~~
~~$A \to A\,1 \mid \epsilon$~~ $A \to A\,1\,F \mid \epsilon$
~~$F \to 0\,A\,1 \mid \epsilon$~~

By the addition of F we have introduced a new variable to make the language string (Format $A \to \alpha A$).

c. Parse tree justification



3) $(ab) + aa.(ab)^{*}a.$

$S \to A\,aa\,A \mid \epsilon$
$A \to aA \mid bA \mid \epsilon$

$S \to A.aa\,A.$
$S \to .aA.aa\,A$
$S \to abAaaA$
$S \to ababaaA$ .... So on.

$\Rightarrow ababaaaba$ → Getting this from the CFG above.

$S \to AaaA.$
$\to aAaaA$
$\to abAaaA$
$\to abaAaaA$
$\to ababAaaA.$ $(\epsilon \to A.)$
$\to abab.aaA.$
$\to .ababaaaA.$
$\to ababaaabA.$
$\to ababaaabaA$ (ϵA)
$\to ababaaaba.$

4. Context free grammar for strings of 0's & 1's.

(i) Atleast three 1's anywhere in the string.

Regular expression = $0^*1 0^*1 0^*1 (0+1)^*$

$L = \Sigma(0,1)$.

Context free grammar :-

$S \rightarrow A1A1A1B$

$A \rightarrow 0A \mid \epsilon$

$B \rightarrow 0B \mid 1B \mid \epsilon$.

(ii) Strings of 0's & 1's whose string is always odd with 0 in m

Regex for this :- $((0+1)(0+1))^*(0+1)\,0\,((0+1)(0+1))^*(0+1)$

Language $L = \Sigma(0,1)$

Context free grammar

$S \rightarrow A0A$.

$A \rightarrow 0 \mid A \mid \epsilon$.

(iii) $a^i b^j c^k$ (with $i = j$ or $j = k$) $(i, j, k \geq 0) \rightarrow \in$ empty string

$S_1 \Rightarrow AB \mid \in$
$A \rightarrow aAb \mid ab \mid \in$
$B \rightarrow Bc \mid c \mid \in$

$\rightarrow$ Ans.

$S_2 \Rightarrow A_1 B_1 \mid \in$
$A_1 \rightarrow aA_1 \mid a \mid \in$
$B_1 \rightarrow bB_1 c \mid bc \mid \in$

Grammar = $S \rightarrow S_1 \mid S_2$
$S_1 \rightarrow AB \mid \in$
$A \rightarrow aAb \mid ab \mid \in$ ← Confusing the terms could become of or.
$B \rightarrow Bc \mid C \mid \in$
$S_2 \rightarrow A_1 B_1 \mid \in$
$S_2 \rightarrow bB_1 c \mid bc \mid \in$

(iv) $\{a^i b^j c^k\} : \{i + j = k,\ i, j, k \geq 0\} \in$ empty string

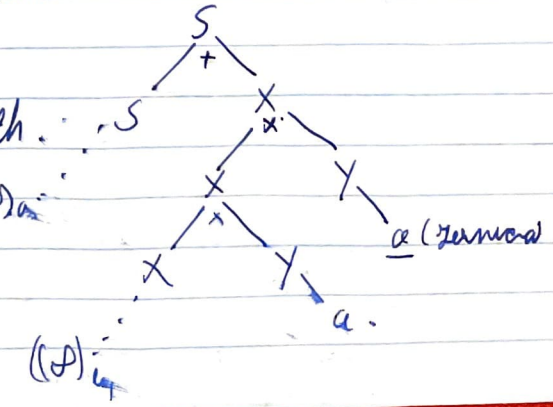
$S \rightarrow aSc \mid A$
$A \rightarrow bAc \mid \in$

$L = \{abcc, aabbcc, \ldots\}$
No of a + No of b = No of c.

5.

$S \rightarrow S + X \mid X$
$X \rightarrow X * Y \mid Y$
$Y \rightarrow a$

(a) In short, left recursion with a top down approach will always end in an infinite tree
Example



As you can see that both the branches will end towards (S) i.e. S.

(b) In order to eliminate the left recursion, we need to re-write this grammar.

we can make it Right recursive on a top down approach.

$S \rightarrow S + X \mid X$ $\Rightarrow$ $S \rightarrow X S'$

$S' \rightarrow \in \mid + T S'$

rest Remain the same

$X \rightarrow x * Y \mid Y$

$Y \rightarrow a$ (which is a terminal.

$[S \rightarrow \alpha . S]$ → Right Sc.

Ques 3 Regular expression

2. Closure property {3,5}

In the example explained above Ax can be converted to A A A ... x times continuously. Moreover A{x,y} can only be done (A x times | A x+1 times ... A .. y times) so these expressions by nature are regular expressions. and hence they are included in the closure properties.

A{x,y} → (A x... | A x.. +1 | A x.... y times).

So these can be deduced to a regular expression & hence. they can be included in the closure properties

1(a) (b+c)* a{1} (b+c)*.

[bc]* a{1} [bc]*.

(b) L→ ..bcabaa bbb .... → (b+c)* aaa . (b+c)*.

→ [bc.]* a{3}* [bc]*

(d) L→ 202, 42, 58. (even number)

→ "\d* [02468] $" \d → integers from [0-9]

(c) String of 5.

\b \d {5} \b. \d → integer [0 - 9]
\b → word boundary.

(e). Regular expression for a string without 101.

→. ^0*(1|00+)*0*$

(f) (0[1-9]|1[012])[-\/.](0[1-9]|[12][0-9]|3[01])[-\/.](19|20)
\d\d.

\d → [0-9] integers

Q5. 1. $5 \times 2(-6+7)/7$

$5 \times 6 + 1)/7$

$5 \times 3/7 = \frac{15}{7}$

2. $8 \times 8/(4+4)/2 \times 2.$

$8 \times 8/8/2 \times 2.$

$64/8/4.$

$64/2.$

32

3. $(5-3) \times 5 \% (8+2.)$

$2 \times 5 \% 5.$

$10 \% 5 = 0$

4. $10(-2.+6) \times (10-2)/(6.+4)$

$(10+4) \times .8/10$

$14 \times .8/10.$

$\frac{112}{10} = 11.2.$

5. $(2+5)/2 \times (5-4) \% 2.$

$7/2. \times 1 \% 2.$

$\frac{7}{2} \times 1. = 3.5$

6. $(12-4) \times (9-3)/10/6.$

$(8 \times .5)/10/6.$

$8 \times 5/10/6.$

$8 \times 5/1.66$

8×3.012

$= 24.096.$

Ques 6 - Also shared. is a text file.

```
if (f() && h() && i() || g() f() && i())
  {
    cout << "What lovely weather" << endl;
  }
  bool f()
  {
     cout << "Hello";
     return true;
  }
  bool g()
  {  cout << "World!";
     return false;
  }
  bool h()
  {  cout << "There";
     return false;
  }
  bool i()
  {  cout << "Darling!" << endl;
     return false;
  }
```

a) Yes., C++ compilers are required to implement the short circuit evaluation
This is mentioned about both the operators in section 23.15.8

c) Yes, in the case of Java the short circuit evaluation is done as well. Specific answer is given in section 15.24

7.

Unit	Var	where Declared
Main	a	main
	b.	main.
Sub 1.	a	Sub 1.
	b.	Sub 1.
Sub 2	a	Sub 2.
	b	Sub 2
	c.	Sub 2.
Sub 3	a	Sub 2.
	b	Sub 3
	c	Sub 2.
	d	Sub 3
	e.	Sub 3