Programming Language, Home work-1

Tulsi Jain

1. C and C++ Language Standards

1.1

Yes, It would legal as /* denotes the starting of multi-line comment hence // (single line comment) would be simply ignored as a part of multi-line comment. Neither "//" does not hold any meaning in multi line comment nor it would through an error.
Reference: 2.7 Comments, Standard ECMA-334 C

1.2

C#: A function declared as virtual can be overridden.
Reference: 15.6.5 Function member invocation, Standard ECMA-334 C
C++: A function not declared as virtual can't be overridden.
Reference: 10.3 Virtual functions, C++ International Standard, 2014

1.3

Five basic elements of lexical structure analysis are Line terminators, White space, Comments, Tokens, Prepossessing Directives.
Tokens are used by the parser tree. Reference: 7.3 Lexical analysis, Standard ECMA-334 C

1.4

It means public and protected members of Class B (base) would be accessible as protected members of class A. And private members of Class B would not be accessible at all. Reference: 10 Derived classes, C++ International Standard, 2014

1.5

Access modifier: New, Public, Protected, Internal, Private, Abstract, Sealed, Static
Default access modifier: internal
Reference: 15.2.2 Class modifiers Standard ECMA-334 C

1.6

The default constructor, copy constructor, copy assignment operator, move constructor, move assignment operator and destructor are six methods provided by the constructor.
Reference: 12 Special member functions, C++ International Standard, 2014

1.7

Control can't be transferred to either try into a try block or into a catch clause. Though it is legal to jump from a try block or catch handler to outside the block or handler. In this case, each variable declared in the try block will be destroyed in the context that directly contains its declaration.
Reference: 15.3 Exceptions Handling, C++ International Standard, 2014

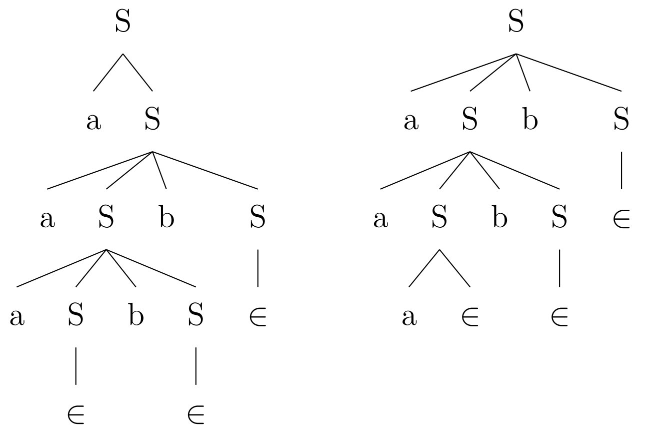
2. Grammars

2.1

Given Grammar

$\langle S\rangle$ ::= ‘a’$\langle Si\rangle$
  | ‘a’$\langle Si\rangle$ ‘b’$\langle Si\rangle$
  | $\in$

Let’s consider the string **aaabb**



Two parser tree exists for same string hence grammar is ambiguous.

2.2

Given Grammar

$\langle E\rangle$ ::= $\langle E\rangle$ ‘+’ $\langle T\rangle$
  | $\langle T\rangle$

$\langle T\rangle$ ::= $\langle T\rangle$ ‘*’ $\langle F\rangle$
  | $\langle F\rangle$

$\langle F\rangle$ ::= $\langle E\rangle$
  | **id**

Let’s consider the string **id**

$$\begin{tikzcd}
E \arrow[d, dash] & & E \arrow[d, dash] \\
T \arrow[d, dash] & & T \arrow[d, dash] \\
F \arrow[d, dash] & & F \arrow[d, dash] \\
\mathrm{id} & & E \arrow[d, dash] \\
 & & T \arrow[d, dash] \\
 & & F \arrow[d, dash] \\
 & & \mathrm{id}
\end{tikzcd}$$

Two parser tree exists for same string hence grammar is ambiguous.

2.3

$\langle S \rangle$ ::= aSa
  | bSb
  | cSc
  | $\langle$*empty*$\rangle$

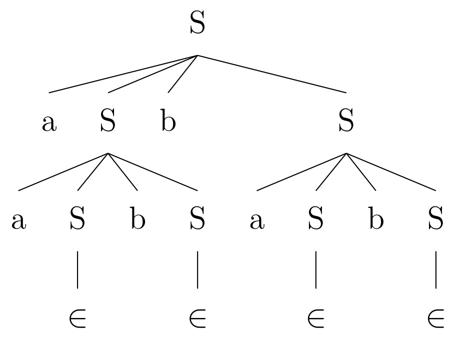
2.4

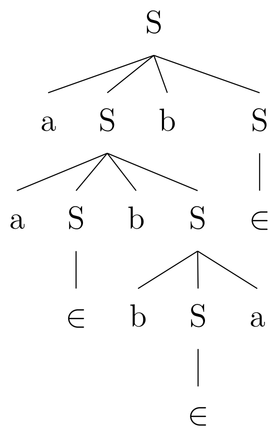
2.4.1

$\langle S \rangle$ ::= aSbS
  | bSa
  | $\langle$*empty*$\rangle$

Show derivation for string **aabbab**

2.4.2





For first tree Left tree derivation
S → aSbS → aaSbSbS → aabSbS → aabbS → aabbaSbS → aabbabS → aabbab

For second tree Right tree derivation
S → aSbS → aSb → aaSbSb → aaSbbSab → aaSbbab → aabbab

Two parser tree exists for same string hence grammar is ambiguous.

2.4.3

Yes, Grammar is ambiguous. Revised grammar

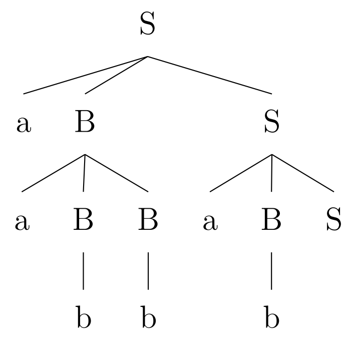
$\langle S \rangle$ ::= aBS
  | bAS
  | $\langle \textit{empty} \rangle$

$\langle B \rangle$ ::= b
  | aBB

$\langle A \rangle$ ::= a
  | bAA

2.4.4

No, it is not not. Only one parse tree exist for **aabbab**.



2.5

$\langle S \rangle$ ::= aSa
  | aSb
  | bSa
  | bSb
  | a

3. Regular expressions

3.1

```
    ^\d*1+\d*$
```

3.2

```
    ^(?:.{5})+$
```

3.3

```
    ^(\d+(\d*|\.\d+)(\d*|E(\+|\-|\.)\d+))$
```

3.4

```
A = ^(((0[1-9]|1[012])[\-](0[1-9]|1[0-9]|2[0-8]))|((0[13578]|1[02])[\-](29|30|31))|
((0[4,6,9]|11)[\-](29|30)))[\-](19|[2-9][0-9])\d\d$
B = ^02[\-]29[\-](19|[2-9][0-9])(00|04|08\12|16|20|24|28|32|36|40|44|48|52
|56|60|64|68|72|76|80|84|88|92|96)$
 Final Answer is A|B (Year Starting from 1900), while grading copy each line individually.
```

3.5

```
    ^(0|2|4|6|8)*$|^(((0|2|4|6|8)*((1|3|5|7))+){2})*$
```

4. JavaScript jQuery Parser

Look for folder names question 4

5. Associativity and Precedence

5.1

$\Longrightarrow 2 + 9 - 5 + 3 - 8$
$\Longrightarrow 11 - 5 + 3 - 8$
$\Longrightarrow 6 + 3 - 8$
$\Longrightarrow 9 - 8$
$\Longrightarrow 1$

5.2

$\Longrightarrow 5 \times 2 \ / \ 5 \times 6$
$\Longrightarrow 5 \times 2 \ / \ 30$
$\Longrightarrow 5 \ / \ 15$
$\Longrightarrow 1/3$

5.3

$\Longrightarrow 5 - 3 \times 5 \ \% \ 3 + 2$
$\Longrightarrow 2 \times 5 \ \% \ 5$
$\Longrightarrow 2 \times 5 \ \% \ 5$
$\Longrightarrow 2 \times 0$
$\Longrightarrow 0$

5.4

$\Longrightarrow 8 + 6 \ / \ 2 \times 3 + 4$
$\Longrightarrow 14 \ / \ 2 \times 7$
$\Longrightarrow 14 \ / \ 2 \times 7$
$\Longrightarrow 14/14$
$\Longrightarrow 1$

5.5

$\Longrightarrow 1 + 2 \ / \ 3 + 4 - 5 + 6 \ \% \ 7$
$\Longrightarrow 3 \ / \ 7 - 5 + 6 \ \% \ 7$
$\Longrightarrow 3 \ / \ 2 + 6 \ \% \ 7$
$\Longrightarrow 3 \ / \ 8 \ \% \ 7$

$\Longrightarrow 3\ /\ 1$
$\Longrightarrow 3$

5.6

$\Longrightarrow 12 \times 4 - 3\ /\ 10\ /\ 5$
$\Longrightarrow 12 \times 1\ /\ 10\ /\ 5$
$\Longrightarrow 12\ \ 1\ /\ 2$
$\Longrightarrow 12\ /\ 2$
$\Longrightarrow 6$

6. Short-Circuit Evaluation

6.1

For, function, f() $->$ False
For, function, g() $->$ Either
For, function, h() $->$ True
For, function, i() $->$ False

6.2

Yes,
C++ compilers required to implement short-circuit evaluation. Default OR and And operator are || and && respectively in C++. Howover, If user defined operator are used that compiler does not guarantee the short-circuit evaluation. This is reason which makes overloading of OR and And operator a bad thing in C++.
Reference: 5.14 Logical AND operator, 5.15 Logical OR operator, C++ International Standard, 2014

6.3

Yes,
15.7 Evaluation Order Java programming language guarantees that the operands of operators appear to be evaluated in a specific evaluation order, namely, from left to right. Section **15.7.2 Evaluate Operands** before Operation also states that every operand of an operator is fully evaluated before any part of the operation itself is performed(except the conditional operators &&, $||, and$:)
Reference: Section 15.7 and 15.7.2, The Java Language Specification

7. Bindings and Nested Subprograms

Unit	Var	Where Declared
main	a	main
	b	main
sub1	a	sub1
	b	main
	c	sub1
sub2	a	main
	b	sub2
	c	sub2
	d	sub2
sub3	a	sub3
	b	sub3
	c	sub2
	d	sub2
	e	sub3