MADE EASY
India's Best Institute for IES, GATE & PSUs

Harsh Kasyap (Enroll User) | Log Out

HOME
MY TEST
MY PROFILE
REPORTS
BUY PACKAGE
NEWS
Ask an Expert

SCORE CARD | TIME MANAGEMENT | QUESTIONS REPORT | SOLUTION | COMPARE YOUR SELF

Solution Report For Compiler Design-1

Represent whole test solution with correct and incorrect answers.

View Your Test Analysis

Q. No	Question Status

Q.1

Consider the following SDT, where $(A \cdot i) = f(S \cdot i)$

Q_1: $S \rightarrow AB$ where, $i = f(S \cdot i)$
$B.i = f(A \cdot s)$
$S \cdot s = f(B \cdot s)$

Q_2: $P \rightarrow QR$
$R \cdot i = f(P \cdot i)$
$Q \cdot i = f(R \cdot s)$

Here $\cdot i$ and $\cdot s$ corresponds to inherited and synthesized attributes respectively. Which of the following is correct?

a. Q_1 is L-attributed but not Q_2
b. Q_2 is L-attributed but not Q_1
c. Both are L-attributed
d. None of them is L-attributed

Attempt | Correct **Correct Ans.** | a | FAQ? | Have any doubt?

Solution. 1

(a)

Let the production be, $X \rightarrow Y_1 Y_2 \ldots Y_i Y_{L+1} \ldots Y_j$

SDD is L-attributed iff:

Inherited attributes of Y_L:

1. Should only be derived from any attribute of $\{Y_1 \ldots Y_{i-1}\}$
2. Only inherited attributes of X.

Q.2 **Which of the following statements is correct?**

a. In a compiler, the module that checks every character of the source text is called symbol table.
b. In a compiler the keywords of a language are recognized during dataflow analysis.
c. An interpreter can give better error diagnostic than a compiler.
d. The loader resolves external memory references, where the code in one file may refer to a location in another file.

Attempt | **Incorrect** **Your Ans.** **b** **Correct Ans.** **c** | FAQ? | Have any doubt?

Solution. 2

(c)

• In a compiler the module that checks every character of the source text is called lexical analysis.
• In a compiler, the keywords of a language are recognized during lexical analysis of the program.
• An interpreter can give better error diagnostic than a compiler beccause it executes the source program statement by statement.
• The linker resolves external memory references, where the code in one file may refer to a location in another file.

Q.3 **Consider the following statements regarding run-time environment**

Which of the above statement is true?

a. The storage used for heap section can grow at run time but not stack section.
b. Only control links and access links are saved or restored when a function call or return happen at runtime.
c. Control links are used in the activation record to access the non-local data.
d. Temporary variables are one of the contents of an activation record.

Attempt Incorrect **Your Ans.** a **Correct Ans.** d FAQ? Have any doubt?

Solution. 3

(d)
• Both stack as well as heap section can grow at run-time.
• Program counter and register values are also needed apart from control and access links while function calls.
• Control links are used to point to the activation record of the caller while function calls.
• Temporary variables are a part of activation record.

Q.4 **Consider the following statements:**

S_1 : It is easy to design compiler for different source languages and target machines because of the two phase division of compiler.

S_2 : Compiler uses a stack that contain a record for each variable name, with fields for the attributes of the name.

Which of the following is true with respect to above statement?

a. only S_1
b. Only S_2
c. Both are true
d. None is true

Attempt Correct **Correct Ans.** a FAQ? Have any doubt?

Solution. 4

(a)

• The compiler is divided into two phases front end phase and Back-end phases. All the compiler modules from lexical analysis till ICG are in the front phase and the modules after ICG till code generation are in back end. Hence, compilers can be generated by combining front end with back end for different machines.

• The data structure used is symbol table that contain a record for each variable name, with fields for the attributes of the name.

Q.5

Consider the following program segment input to the compiler

```
main ( )
{
    int * a, b;
    a = 10;
    a = &b;
    printf ("%d%d", b, * a);
    b = */* pointer * /b;
}
```

The number of tokens in the above C code are _____.

Attempt | **Incorrect** **Your Ans.** **34** | **Correct Ans.** **35** | FAQ? | Have any doubt?

Solution. 5

35

```
main ( )
{
  int * a , b ;
  a = 10 ;
  a = & b ;
  printf ( "%d%d" , b , * a ) ;
  b = * / * pointer * / b ;
}
```

Q.6

Let G be a grammar with the following productions.

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$T \rightarrow (E) \mid T - F$

$F \rightarrow id$

If LR(1) Parser is used to construct the DFA using the above productions, then how many look-a-heads are present for an item $T \rightarrow \bullet T * F$ in the initial state?

Attempt | **Incorrect** **Your Ans.** **2** | **Correct Ans.** **4** | FAQ? | Have any doubt?

Solution. 6

4

$E' \rightarrow \bullet\ E, \$$ $\Rightarrow$ 1 Look-a-heads
$E \rightarrow \bullet\ E + T, \{\$, +\}$ $\Rightarrow$ 2 Look-a-heads
$E \rightarrow \bullet\ T, \{\$, +\}$ $\Rightarrow$ 2 Look-a-heads
$T \rightarrow \bullet\ T * F, \{*, -, \$, +\}$ $\Rightarrow$ 4 Look-a-heads
$T \rightarrow \bullet\ F, \{*, -, \$, +\}$ $\Rightarrow$ 4 Look-a-heads
$T \rightarrow \bullet\ (E), \{*, -, \$, +\}$ $\Rightarrow$ 4 Look-a-heads
$T \rightarrow \bullet\ T - F, \{*, -, \$, +\}$ $\Rightarrow$ 4 Look-a-heads
$T \rightarrow \bullet\ id, \{*, -, \$, +\}$ $\Rightarrow$ 4 Look-a-heads

Q.7

Consider the following SDT

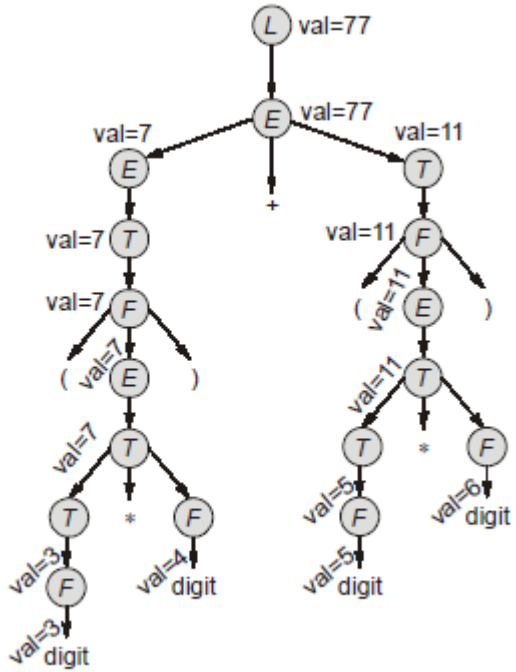
$L \rightarrow E$	$\{L.val = E.val\}$
$E \rightarrow T$	$\{E.val = T.val\}$
$E \rightarrow E + T$	$\{E.val = E_1.val * T.val\}$
$T \rightarrow F$	$\{T.val = F.val\}$
$T \rightarrow T * F$	$\{T.val = T_1.val + F.val\}$
$F \rightarrow (E)$	$\{F.val = E.val\}$
$F \rightarrow digit$	$\{F.val = digit.val\}$

The value of input expression "(3 * 4) + (5 * 6)" is _____.

Attempt | **Correct** **Correct Ans.** | **77** ❓ FAQ? | ❓ Have any doubt?

Solution. 7

77



Q.8 **Match List-I with List-II and select the correct answer using the codes given below the lists:**

List-I

A. Puts together all of the executable object files into memory for execution.

B. Linking relocatable machine code with other relocatable object files into code that runs on machine.

C. It takes the output generated by the compiler as input and generate relocatable machine code as the output.

D. Expanding macros into source language statements.

List-II

1. Linker

2. Loader

3. Assembler

4. Preprocessor

Codes:

	A	B	C	D
(a)	1	2	3	4
(b)	2	1	3	4
(c)	3	4	2	1
(d)	4	3	1	2

a. a
b. b
c. c
d. d

Attempt | **Correct** **Correct Ans.** | **b** | FAQ? | Have any doubt?

Solution. 8

(b)
Considering the complete language-processing system. The preprocessor expand macros, into sourc language statements and generate modified source program as the output. The modified source program is then fed to a compiler which generates assembly level program. The assembly language is processed by a program called an assembler that produces relocatable machine codes. the linker resolves the linking between various relocatable object files and library files into the code that actually runs on the machine. Then the loader puts together all of the executable object files into memory for execution.

Q.9 **Consider the C program given below:**

```
main ( )
{
    a = a + b;
    c = a * c;
    d = c – d;
    a = c / d ;
    printf("%d", a);
}
```

What will the minimum number of nodes and edges present in the DAG representation of the output of above C program?

a. 6 and 6
b. 7 and 6
c. 6 and 7
d. 8 and 8

Attempt | **Correct** **Correct Ans.** | **d** | FAQ? | Have any doubt?

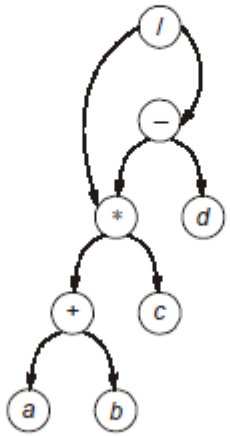
Solution. 9

(d)

$a = (a + b)$
$c = a * c$ } $c = (a + b)*c$

$d = (c - d)$
$a = c / d$ } $a = c /(c - d)$

Final expressions is : $a = (a + b)*c /((a + b)*c - d)$ so DAG representation for above expressi



Number of nodes = 8
Number of edges = 8

Q.10

Let 'G' be a grammar with the following translations:

$S \rightarrow p\{\text{print "G"}\}\ P$
$P \rightarrow q\{\text{print "A"}\}\ Q$
$P \rightarrow r\{\text{print "T"}\}$
$P \rightarrow \varepsilon\{\text{print "E"}\}$
$Q \rightarrow s\{\text{print "A"}\}\ P$
$Q \rightarrow \varepsilon\{\text{print "O"}\}$

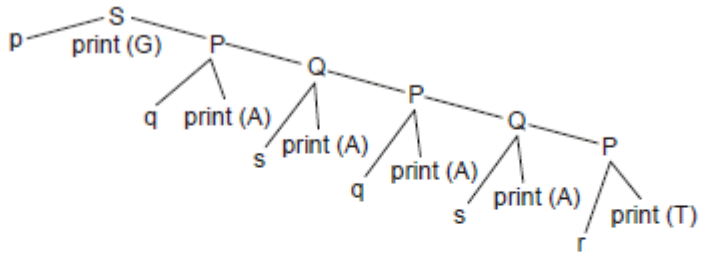
What is the output produced for the input "pqsqsr" using the bottom-up parsing with above translations?

a. TAAAG
b. TAAAAG
c. GAAA
d. AAAGAT

Attempt | Correct | Correct Ans. | b | FAQ? | Have any doubt?

Solution. 10

(b)



Since, the parser is a bottom-up parser, which is reverse of the right most derivation of the string. evaluating from right to left, bottom to top, the output produced will be "TAAAAG".

Q.11 **Consider the syntax directed translation scheme below for the grammar:**

$$N \rightarrow L$$
$$L \rightarrow L + B \mid B$$
$$B \rightarrow 0 \mid 1$$

The schema is

$N \rightarrow L$	{N.val = L.val}
$L \rightarrow L + B$	{L.val = L.val * 2 + B.val}
$L \rightarrow B$	{L.val = B.val}
$B \rightarrow 0$	{B.val = 0}
$B \rightarrow 1$	{B.val = 1}

The value of N.val is

a. The number of bits in the input string

b. The value of the non-decimal input in binary form
c. The value of the binary input in decimal form
d. None of the above

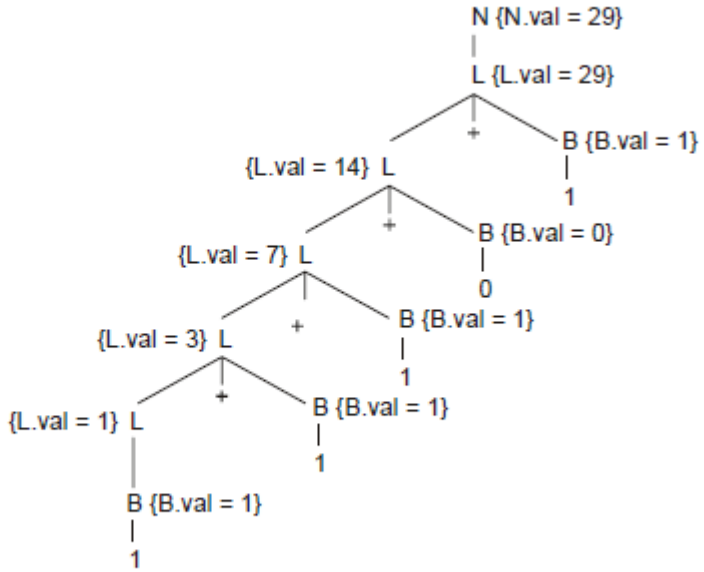
Attempt Incorrect **Your Ans.** d **Correct Ans.** c

FAQ? Have any doubt?

Solution. 11

(c)

Consider the input string '11101'.



'11101' corresponds to the decimal value 29.

Q.12 **Which of the following is correct?**

a. One of the purposes of using intermediate code in compilers is to improve the register allocation.
b. Leaf node of the tree always have only inherited atrribute.

c. The difference between assembly code generation and intermediate code generation lies in the number of registers used by both of them to hold the intermediate results.

d. The output of intermediate code generation is machine dependent.

Attempt Incorrect **Your Ans.** b **Correct Ans.** c FAQ? Have any doubt?

Solution. 12

(c)

• Intermediate code generation enhances the portability of the code.

• Leaf node of the tree have synthesized attributes as the values are taken from symbol table.

• ICG can use any number of registers to hold the intermediate code. Later, code optimization is performed to generate the assembly code.

• The output of intermediate code generation is machine independent.

Q.13

Consider the following expression:

$$((x + y) - ((x + y) * (x + y))) + ((x + y) * (x + y)) + ((x + y) \div (x + y))$$

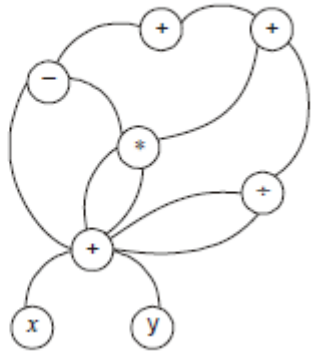
The number of nodes to represent the DAG for the above expression is _____.

Attempt Correct **Correct Ans.** 8 FAQ? Have any doubt?

Solution. 13

8

DAG for the expression is:



Q.14 **Consider the following productions along with their semantic rules:**

Production	Semantic Rules
$T \rightarrow FT'$	$T'.i = F.val$ $T.val = T'.s$
$T' \rightarrow *FT_1'$	$T_1'.i = T'.i + 2 * F.val$ $T'.s = T_1'.s$
$T' \rightarrow \in$	$T'.s = T'.i$
$F \rightarrow id$	$F.val = id.lexval$

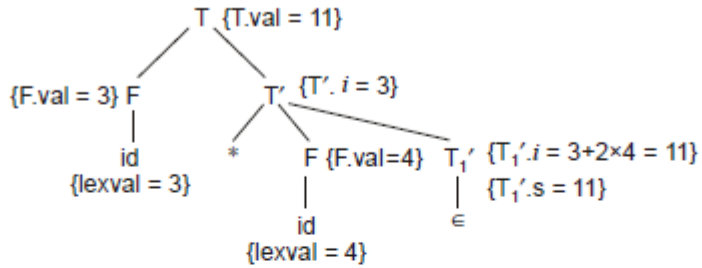
Here · *i* and · *s* corresponds to inherited and synthesized attributes respectively. The value for expression 3 * 4 _____.

Attempt Correct **Correct Ans.** 11 FAQ? Have any doubt?

Solution. 14

11

Constructing annotated parse tree:



F → id: id·lexval is the lexval returned by the lexical analyzer, F· val =3.
Similary F· val = 4 in other subtree.
T → FT′: T′ takes the value F·val i.e., $T' \cdot i$ = F·val = 3. Left operand is transfered to right subtree.
T′ → *FT$_1$′: $T_1' \cdot i$ takes the value $T_1' i + 2 * \text{F·val} = T_1' \cdot i = 11$
T′ → ∈: $T' \cdot S = T' \cdot i = 11$
T → FT′: $T \cdot S = T_1' \cdot S = 4$
T → FT′: T·val = T·syn = 11
3 * 4 is evaluated to 11.

Q.15

Consider the following expression.

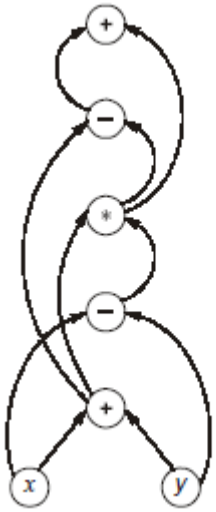
$$((x + y) - ((x + y) * (x - y))) + ((x + y) * (x - y))$$

The number of nodes to represent the DAG for above expression is _____.

Attempt | **Correct** **Correct Ans.** | **7** | FAQ? | Have any doubt?

Solution. 15

7



Q.16 **Consider the following grammar :**

$S \rightarrow Aa \mid B$

$B \rightarrow a \mid bC$

$C \rightarrow a \mid \in$

The number of productions in simplified CFG is _____.

Attempt | **Incorrect** **Your Ans.** **3** | **Correct Ans.** **8** | FAQ? | Have any doubt?

Solution. 16

8

$$S \to Aa \mid B$$
$$B \to a \mid bC$$
$$C \to a \mid \in$$

First remove null production i.e. $C \to \in$, then resultant grammar is

$$S \to Aa \mid B$$
$$B \to a \mid bC \mid b$$
$$C \to a$$

Second remove unit production i.e. $S \to B$, then resultant grammar is

$$S \to Aa \mid a \mid bC \mid b$$
$$B \to a \mid bC \mid b$$
$$C \to a$$

There is no useless production so simplified CFG is

$$S \to Aa \mid a \mid bC \mid b$$
$$B \to a \mid bC \mid b$$
$$C \to a$$

So, the number of production is 8.