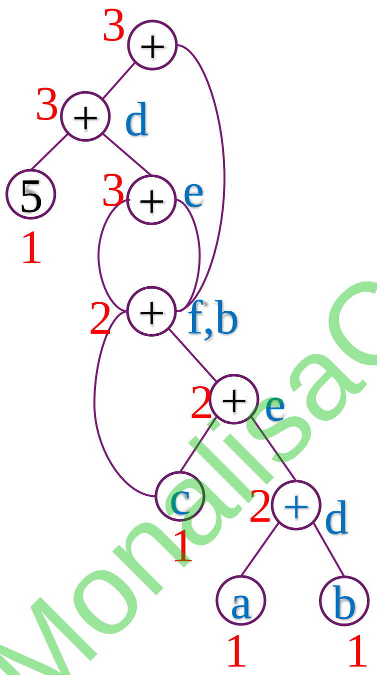
Compiler Design
Chapter 3:SDT,RTE

GATE CS PYQ
by Monalisa

- GATE CS 2010,Q14: Which languages necessarily need heap allocation in the runtime environment?
- **(A)** Those that support recursion
  (B) Those that use dynamic scoping
  (C) Those that allow dynamic data structures
  (D) Those that use global variables
- (A) Those that support recursion need Stack allocation
- (B) The scope of a declaration of x is the region of the program in which uses of x refer to this declaration.
- With dynamic scope, as the program runs , the same use of x could refer to any of several different declarations of x.
- (C)Heap allocation is needed for dynamic data structures like tree, linked list, etc. So the languages which allow dynamic data structure require heap allocation at runtime.
- (D) Those that use global variables need static allocation.
- Ans : **(C)** Those that allow dynamic data structures

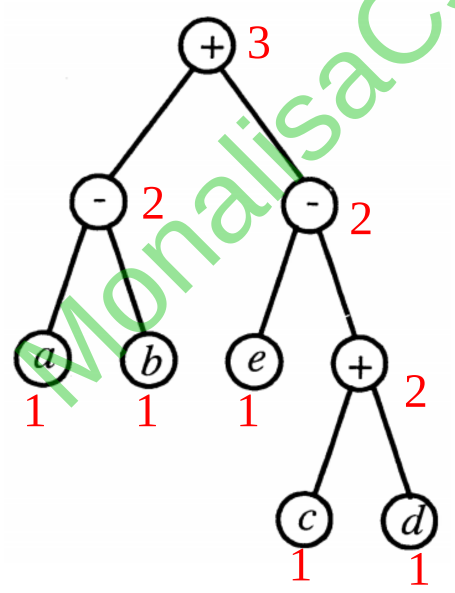
- GATE CS 2010,Q37: The program below uses six temporary variables a, b, c, d, e, f.
- a = 1
- b = 10
- c = 20
- d = a+b
- e = c+d
- f = c+e
- b = c+e
- e = b+f
- d = 5+e
- return d+f



- Assuming that all operations take their operands from registers, what is the minimum number of registers needed to execute this program without spilling?
- **(A)** 2 **(B)** 3 **(C)** 4 **(D)** 6

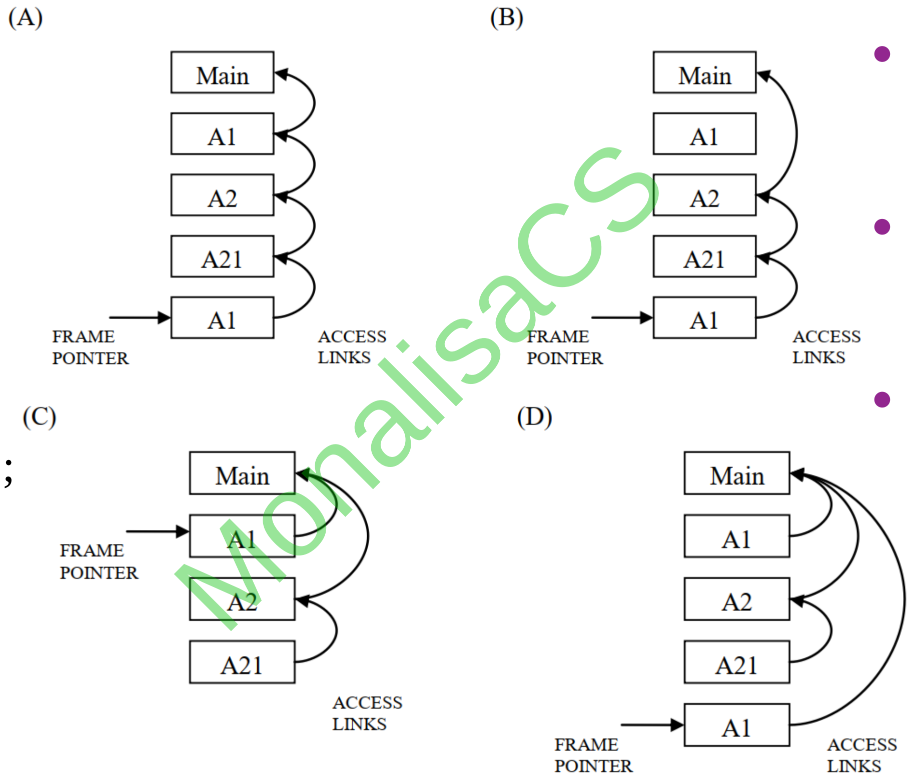
- LD R1,a //R1=a=1
- LD R2,b //R2=b=10
- LD R3,c //R3=c=20
- ADD R1,R1,R2 //R1=R1+R2
- ADD R1,R3,R1 //R1=R3+R1
- ADD R2,R3,R1 //R2=R3+R1
- ADD R2,R3,R1 //R2=R3+R1
- ADD R1,R2,R2 //R1=R2+R2
- ADD R3,#5,R1 //R3=5+R1
- ST d,R3
- ST f,R2
- Minimum 3 registers needed.
- Ans: **(B)** 3

- GATE CS 2011,Q36: Consider evaluating the following expression tree on a machine with load-store architecture in which memory can be accessed only through load and store instructions. The variables a, b, c, d and e initially stored in memory. The binary operators used in this expression tree can be evaluate by the machine only when the operands are in registers. The instructions produce results only in a register. If no intermediate results can be stored in memory, what is the minimum number of registers needed to evaluate this expression?
- **(A)** 2
  (B) 9
  (C) 5
  (D) 3
- Ans: **(D)** 3



- GATE CS 2012,Q36: Consider the program given below, in a block-structured pseudo-language with lexical scoping and nesting of procedures permitted.
- Program main;

```
        Var ...
        Procedure A1;
          Var ...
        Call A2;
        End A1
        Procedure A2;
          Var ...
          Procedure A21;
                   Var ...
                   Call A1;
          End A21
          Call A21;
        End A2
        Call A1;
End main.
```



- A1,A2 are defined under Main.
- So A1,A2 Access link are pointed to main.
- A21 is defined under A2 hence its Access link will point to A2 .
- Ans: (D)

- Consider the calling chain: Main → A1 → A2 → A21 → A1
- The correct set of activation records along with their access links is given by

- GATE CS 2013,Q48: **Common Data for Questions 48 and 49:** The following code segment is executed on a processor which allows only register operands in its instructions. Each instruction can have atmost two source operands and one destination operand. Assume that all variables are dead after this code segment.
- c = a + b;
- d = c * a;
- e = c + a;
- x = c * c;
- if (x > a) { y = a * a;}
- else { d = d * d;
- e = e * e;}

```
c = a + b;              ADD R2 ,R1,R2 ;
x = c * c;              MUL R2,R2,R2        spill c
if (x > a)
{ y = a * a; }          MUL R2,R1,R1;
else { d = c * a;       MUL R2,R2,R1;
d = d * d;              MUL R2,R2,R2;
e = c + a;              ADD R2,R2,R1 ;
e = e * e; }            MUL R2,R2,R2;
```

- Q.48: Suppose the instruction set architecture of the processor has only two registers. The only allowed compiler optimization is code motion, which moves statements from one place to another while preserving correctness. What is the minimum number of spills to memory in the compiled code?
- (A) 0 (B) 1 (C) 2 (D) 3
- Total number of spills to memory is 1 .only c
- Ans: (B) 1

- GATE CS 2013,Q49:**Common Data for Questions 48 and 49:** The following code segment is executed on a processor which allows only register operands in its instructions. Each instruction can have atmost two source operands and one destination operand. Assume that all variables are dead after this code segment.
- c = a + b;
- d = c * a;
- e = c + a;
- x = c * c;
- if (x > a) { y = a * a;}
- else { d = d * d;
-             e = e * e;}

c = a + b;	ADD R2 ,R1,R2 ;
d = c * a;	MUL R3 ,R2,R1;
e = c + a;	ADD R4 ,R2,R1;
x = c * c;	MUL R2 ,R2,R2
if (x > a)	
{ y = a * a; }	MUL R1,R1,R1;
else {d = d * d;	MUL R3,R3,R3;
e = e * e; }	MUL R4, R4, R4;

- Q.49 What is the minimum number of registers needed in the instruction set architecture of the processor to compile this code segment without any spill to memory? Do not apply any optimization other than optimizing register allocation.
  (A) 3 (B) 4 (C) 5 (D) 6
- Minimum number of registers 4.
- Ans: (B) 4

- GATE CS 2014 Set-1,Q17: Which one of the following is **FALSE**?
  (A) A basic block is a sequence of instructions where control enters the sequence at the beginning and exits at the end.
  (B) Available expression analysis can be used for common subexpression elimination.
  (C) Live variable analysis can be used for dead code elimination.
  (D) x=4*5⇒x=20 is an example of common subexpression elimination.
- (A)True
- (B)True, The primary use of available-expression information is for detecting **global common subexpressions.**
- (C) True, Live variable analysis can be used for **dead code elimination** and **register allocation**.
- (D)False , x=4* 5 ⇒ x=20 is an example of **Constant folding** not common subexpression elimination
- Ans:(D) x=4*5⇒x=20 is an example of common subexpression elimination.

- GATE CS 2014 Set-2,Q18: Which one of the following is NOT performed during compilation?
- **(A)** Dynamic memory allocation
  (B) Type checking
  (C) Symbol table management
  (D) Inline expansion
- (A) Dynamic memory allocation is not performed during compilation, it occurs at run time only.
- (B) Type checking performed during completion semantic analysis phase.
- **(C)** Symbol table management performed during compilation .
- (D) Inline expansion is compiler optimization that replaces a call to a function with the body of that function . Performed during compilation.
- Ans: **(A)** Dynamic memory allocation

- GATE CS 2014 Set-2,Q34:For a C program accessing X[i][j][k], the following intermediate code is generated by a compiler. Assume that the size of an integer is 32 bits and the size of a character is 8 bits.
- t0 = i ∗ 1024
- t1 = j ∗ 32
- t2 = k ∗ 4
- t3 = t1 + t0
- t4 = t3 + t2
- t5 = X[t4]
- Which one of the following statements about the source code for the C program is CORRECT?
- **(A)** X is declared as "int X[32][32][8]".
- **(B)** X is declared as "int X[4][1024][32]".
- **(C)** X is declared as "char X[4][32][8]".
- **(D)** X is declared as "char X[32][16][2]".
- Let a[r][c][b],Row major order
- Loc a[i][j][k]= [i ×c ×b +j ×b+k] ×w if base is 0.

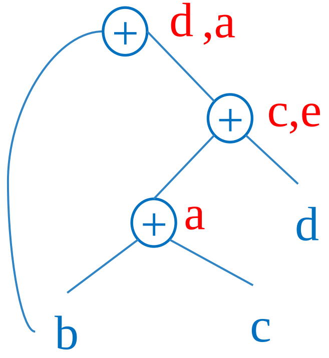
- t5 = X[t4]
- =X[t3 + t2]
- =X[t1 + t0 + k ∗ 4]
- =X[j ∗ 32 + i ∗ 1024 + k ∗ 4]
- =X[i ∗1024 +j∗32+k∗4]
- =X[i ∗ 256+j ∗ 8+ k]*4
- =X[i ∗32*8+j∗8+k]*4
- w=4 ,b=8,c=32
- int X[ ][32][8]
- Ans: (A) "int X[32][32][8]".

- GATE CS 2014 Set-3,Q11: The minimum number of arithmetic operations required to evaluate the polynomial $P(X)=X^5+4X^3+6X+5$ for a given value of X, using only one temporary variable is ______.
- $P(X)=x^5+4x^3+6x+5$
- $=x(x^4+4x^2+6)+5$
- $=x(x(x^3+4x)+6)+5$
- $=x(x(x(x^2+4))+6)+5$
- $=x(x(x(x(x)+4))+6)+5$
- 1. $t = x * x$ $[x^2]$
- 2. $t = t + 4$ $[x^2 + 4]$
- 3. $t = t * x$ $[x^3+4x]$
- 4. $t = t * x$ $[x^4+4x^2]$
- 5. $t = t + 6$ $[x^4+4x^2+6]$
- 6. $t = t * x$ $[x^5+4x^3+6x]$
- 7. $t = t + 5$ $[x^5+4x^3+6x+5]$
- Minimum number of arithmetic operations 7
- Ans : 7

- GATE CS 2014 Set-3,Q17:One of the purposes of using intermediate code in compilers is to
  (A) make parsing and semantic analysis simpler.
  (B) improve error recovery and error reporting.
  (C) increase the chances of reusing the machine-independent code optimizer in other compilers.
  (D) improve the register allocation.
- (A) Intermediate code is after syntax analysis & semantic analysis . False
- (B) Error recovery & reporting done by all phases of compiler . False
- (C) The intermediate code is independent of machine. By converting source code to intermediate code a machine independent code optimizer may be written. Thus increase the chances of reusing the machine-independent code optimizer in other compilers . True
- (D) Register allocation is done in code generation phase . False
- **Ans : (C)**

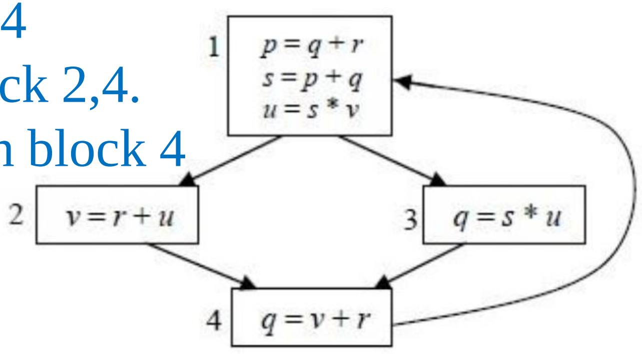
- GATE CS 2014 Set-3,Q18:Which of the following statements are CORRECT?
- 1) Static allocation of all data areas by a compiler makes it impossible to implement recursion.
- 2) Automatic garbage collection is essential to implement recursion.
- 3) Dynamic allocation of activation records is essential to implement recursion.
- 4) Both heap and stack are essential to implement recursion.
- (A) 1 and 2 only (B) 2 and 3 only (C) 3 and 4 only (D) 1 and 3 only
- 1) Static allocation of all data areas by a compiler makes it impossible to implement recursion is true, as recursion requires memory allocation at run time, so it requires dynamic allocation of memory.
- 2) Automatic garbage collection is essential to implement heap not stack .
- 3) Dynamic allocation of activation records is essential to implement recursion .
- 4) Stack is essential to implement recursion not heap.
- Ans : (D) 1 and 3 only

- GATE CS 2014 Set-3,Q34:Consider the basic block given below.
- a = b + c
- c = a + d
- d = b + c
- e = d - b
- a = e + b
- The minimum number of nodes and edges present in the DAG representation of the above basic block respectively are
- **(A)** 6 and 6 **(B)** 8 and 10 **(C)** 9 and 12 **(D)** 4 and 4
- a = b + c
- c = a + d
- d = b + c
- e = d - b = b + c - b = c
- a = d - b + b = d
- DAG representation of above basic block→
- Minimum number of nodes = 6 & edges = 6
- Ans : **(A)** 6 and 6

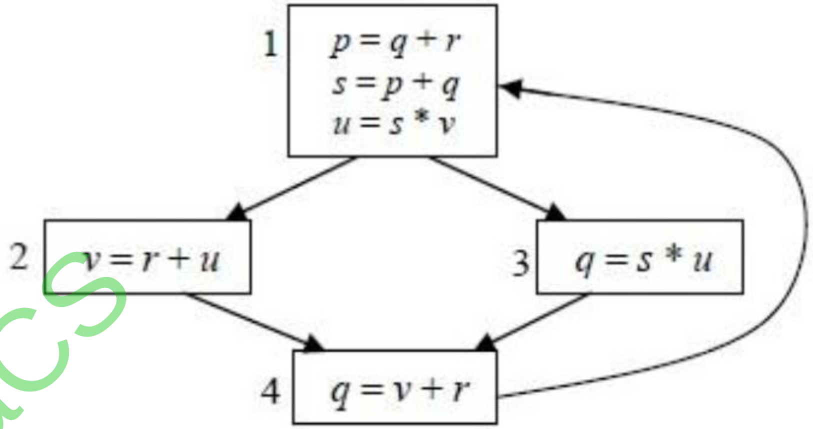


- GATE CS 2015 Set-1,Q50:A variable x is said to be live at a statement S_i in a program if the following three conditions hold simultaneously:
- I. There exists a statement S_j that uses x
- II. There is a path from S_i to S_j in the flow graph corresponding to the program
- III. The path has no intervening assignment to x including at S_i and S_j
- The variables which are live both at the statement in basic block 2 and at the statement in basic block 3 of the above control flow graph are

  (A) p, s, u **(B)** r, s, u **(C)** r, u **(D)** q, v
- **(A)** p is live at block 1 1st-2nd stmt. Dead in block 2,3,4
- s is live in block 1 2nd -3rd stmt & block 3 .Dead in block 2,4.
- u is live in block 1 3rd stmt, block 2 & block 3 .Dead in block 4
- (B) r is live in Block 1,2,3,4
- (C) r is live at block 1,2,3,4 & u is live at block 2, 3.
- r , u both live at block 2& 3.
- (D) q is live in block 4,1 . v is live in block 2,4,1,3.
- **Ans : (C)** r, u



- *OUT[B] =$U_{S \text{ a successor of } B}$ IN[S]*
- *IN[B] = use_B U (OUT[B] - def_B)*
- *use_1 ={q,r,v}, def_1={p,s,u}*
- *use_2 ={r,u}, def_2 ={v}*
- *use_3 ={s,u}, def_3 ={q}*
- *use_4={r,v}, def_4 ={q}*
- *OUT[4] = IN[1] = {q,r,v}*
- *IN [4]={r,v}U[{q,r,v}-{q}]={r,v}*
- *OUT[3] = IN[4] = {r,v}*
- *IN [3]={s,u}U[{r,v}-{q}]={r,s,u,v}*
- *OUT[2] = IN[4]={r,v}*
- *IN [2]={r,u}U[{r,v}-{v}]={r,u}*
- *OUT[1] = IN[2] UIN[3]={r,u}U{r,s,u,v}={r,s,u,v}*
- *IN [1]={q,r,v}U[{r,s,u,v}-{p,s,u}]={q,r,v}*
- The variables which are live both in basic block 2 and 3 are *{r,u}*

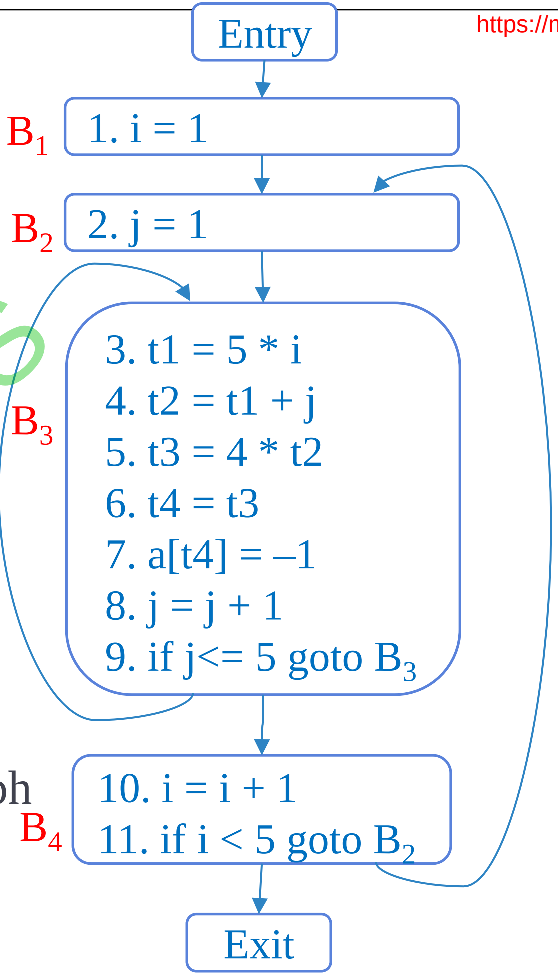


- GATE CS 2015 Set-1,Q55: The least number of temporary variables required to create a three-address code in static single assignment form for the expression $q + r / 3 + s - t * 5 + u * v / w$ is ____.
- t_1=r/3;
- t_2=t∗5;
- t_3=u∗v;
- t_4=t_3/w;
- t_5=q+t_1;
- t_6=t_5+s;
- t_7=t_6−t_2;
- t_8=t_7+t_4;
- We need 8 temporary variables.
- Without SSA form we need 3 temporary variable .
- Ans : 8

- GATE CS 2015 Set-2,Q14:In the context of abstract-syntax-tree (AST) and control-flow-graph (CFG), which one of the following is True?
- **(A)** In both AST and CFG, let node N_2 be the successor of node N_1. In the input program, the code corresponding to N_2 is present after the code corresponding to N_1
  (B) For any input program, neither AST nor CFG will contain a cycle
  (C) The maximum number of successors of a node in an AST and a CFG depends on the input program
  (D) Each node in AST and CFG corresponds to at most one statement in the input program
- (A) In CFG,code of N_2 may be present before N_1 when there is a loop or jump . False
- (B) CFG contains cycle when input program has loop . False
- (C) Successors in AST and CFG depend on input program. True
- (D) In CFG a single node may contain more than one statements . False
- Ans : C

- GATE CS 2015 Set-2,Q29:
- Consider the intermediate code given below:
- 1. i = 1
- 2. j = 1
- 3. t1 = 5 * i
- 4. t2 = t1 + j
- 5. t3 = 4 * t2
- 6. t4 = t3
- 7. a[t4] = –1
- 8. j = j + 1
- 9. if j <= 5 goto(3)
- 10. i = i + 1
- 11. if i < 5 goto(2)
- The number of nodes and edges in the control-flow-graph constructed for the above code, respectively, are
- **(A)** 5 and 7 **(B)** 6 and 7
- **(C)** 5 and 5 **(D)** 7 and 8
- Leader 1,2,3,10

- #nodes=6
- #edges=7
- Ans: **(B)** 6 and 7



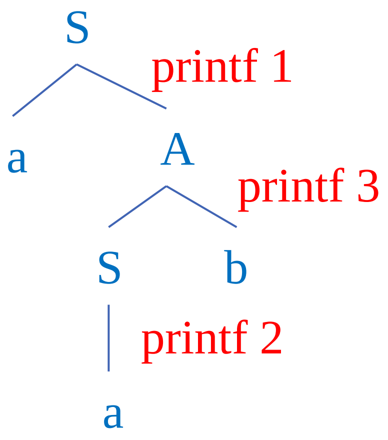
- GATE CS 2016 Set-1,Q19:Consider the following code segment.

```
x = u - t;
y = x * v;
x = y + w;
y = t - z;
y = x * y;
```

The minimum number of *total* variables required to convert the above code segment to *static single assignment* form is________ .

- Static Single Assignment form (SSA) of the given code segment is:
- $x_1 = u - t;$
- $y_1 = x_1 * v;$
- $x_2 = y_1 + w;$
- $y_2 = t - z;$
- $y_3 = x_2 * y_2;$
- Total Variables={x_1, x_2, y_1, y_2, y_3, t, u, v, w, z}
- # total variable = 10
- Ans : 10

- GATE CS 2016 Set-1,Q46: Consider the following Syntax Directed Translation Scheme (SDTS), with non-terminals {S, A} and terminals {**a**, **b**}.
  S → **a**A { print 1 }
  S → **a** { print 2 }
  A → S**b** { print 3 }
  Using the above SDTS, the output printed by a bottom-up parser, for the input **aab** is:(A) 1 3 2 (B) 2 2 3 (C) 2 3 1 (D) syntax error
- S-attributed SDT
- Attributes are evaluated bottom up , post order traversal.
- Ans : (C) 2 3 1



- GATE CS 2016 Set-2,Q19: Match the following:

(P) Lexical analysis	(i) Leftmost derivation
(Q) Top down parsing	(ii)Type checking
(R) Semantic analysis	(iii)Regular expressions
(S) Runtime environments	(iv) Activation records

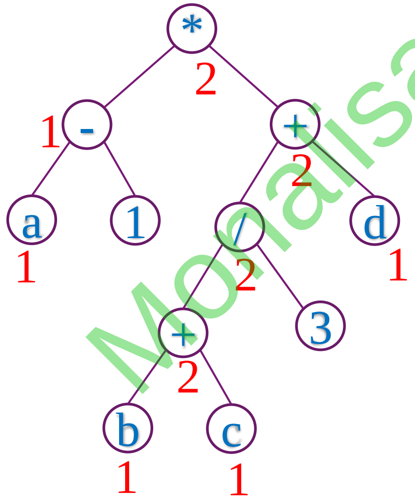
- (A) P ↔ i, Q ↔ ii, R ↔ iv, S ↔ iii (B) P ↔ iii, Q ↔ i, R ↔ ii, S ↔ iv
- (C) P ↔ ii, Q ↔ iii, R ↔ i, S ↔ iv (D) P ↔ iv, Q ↔ i, R ↔ ii, S ↔ iii
- (P) Lexical analysis uses (iii)Regular expressions
- (Q) Top down parsing uses (i)left most derivation.
- (R) Semantic analysis used for (ii) Type checking.
- (S) Runtime environments load (iv)Activation records into stack.
- Ans : (B) P ↔ iii, Q ↔ i, R ↔ ii, S ↔ iv

- GATE CS 2017 Set-1,Q12: Consider the following intermediate program in three address code
- $p = a - b$
- $q = p * c$
- $p = u * v$
- $q = p + q$
- Which one of the following corresponds to a *static single assignment* form of the above code?
- SSA form of the given code:
- $p_1 = a-b$
- $q_1 = p_1 * c$
- $p_2 = u * v$
- $q_2 = p_2 + q_1$
- Ans: B

(A)	(B)	(C)	(D)
$p_1 = a - b$	$p_3 = a - b$	$p_1 = a - b$	$p_1 = a - b$
$q_1 = p_1 * c$	$q_4 = p_3 * c$	$q_1 = p_2 * c$	$q_1 = p * c$
$p_1 = u * v$	$p_4 = u * v$	$p_3 = u * v$	$p_2 = u * v$
$q_1 = p_1 + q_1$	$q_5 = p_4 + q_4$	$q_2 = p_4 + q_3$	$q_2 = p + q$

- GATE CS 2017 Set-1,Q43: Consider the following grammar:
- stmt → **if** expr **then** expr **else** expr; stmt | ε
- expr → term **relop** term | term
- term → id | number
- id → **a** | **b** | **c**
- number → [**0-9**]
- where **relop** is a relational operator (e.g., <, >, ….), ε refers to the empty statement, and **if** ,**then**, **else** are terminals.
- Consider a program P following the above grammar containing ten **if** terminals. The number of control flows paths in P is ________. For example, the program **if** e_1 **then** e_2 **else** e_3
- has 2 control flow paths, $e_1 \rightarrow e_2$ and $e_1 \rightarrow e_3$.
- Every if statement has 2 control flows as given in question.Hence,
- 2 control flow for 1st 'if'
- 2 control flow for 2nd 'if'….
- 2 control flow for 10th 'if'
- Total control flow path=2 × 2 × 2 × .... 10 times = 2^{10} = 1024
- Ans: 1024

- GATE CS 2017 Set-1,Q52: Consider the expression (a-1) * (((b + c) / 3) + d). Let X be the minimum number of registers required by an *optimal* code generation (without any register spill) algorithm for a load/store architecture, in which *(i) only load and store instructions can have memory operands* and *(ii) arithmetic instructions can have only register or immediate operands* .The value of X is ______.
- LD R_1,b
- LD R_2,c
- ADD R_1, R_1, R_2
- DIV R_1,R_1,#3
- LD R_2,d
- ADD R_1, R_1, R_2
- LD R_2,a
- SUB R_2,R_2,#1
- MUL R_1, R_2, R_1
- X=2
- Ans : 2



- GATE CS 2019,Q36: Consider the following grammar and the semantic actions to support the inherited type declaration attributes. Let X_1, X_2, X_3, X_4, X_5 and X_6 be the placeholders for the non-terminals D, T, L or L_1 in the following table:
- Which one of the following are the appropriate choices for X_1, X_2, X_3 and X_4?
- **(A)** $X_1 = L, X_2 = T, X_3 = L_1, X_4 = L$
- **(B)** $X_1 = T, X_2 = L, X_3 = L_1, X_4 = T$
- **(C)** $X_1 = L, X_2 = L, X_3 = L_1, X_4 = T$
- **(D)** $X_1 = T, X_2 = L, X_3 = T, X_4 = L_1$

Production rule	**Semantic action**
$D \rightarrow TL$	X_1.type=X_2.type
$T \rightarrow$ int	T.type=int
$T \rightarrow$ float	T.type=float
$L \rightarrow L_1$,id	X_3.type=X_4.type addType(id.entry,X_5.type)
$L \rightarrow$ id	addType(id.entry,X_6.type)

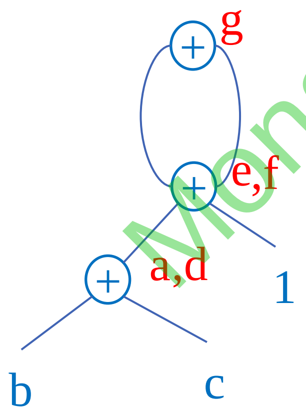
- Inherited type declaration attributes
- $D \rightarrow TL$ {X_1.type=X_2.type}
- L.type=T.type
- X_1=L,X_2=T
- $L \rightarrow L_1$, id {X_3.type = X_4.type }
- L data type inherited from T .
- Now L data type will pass to its children as type is inherited attribute
- L_1.type=L.type
- X_3= L_1,X_4=L
- Ans : **(A)** $X_1 = L, X_2 = T, X_3 = L_1, X_4 = L$

- GATE CS 2020,Q33:Consider the productions A → PQ and A → XY. Each of the five non-terminals A,P,Q,X, and Y has two attributes: s is a synthesized attribute, and i is an inherited attribute. Consider the following rules.
- **Rule 1:** P.i=A.i+2, Q.i=P.i+A.i, and A.s=P.s+Q.s
- **Rule 2:** X.i=A.i+Y.s and Y.i=X.s+A.i
- Which one of the following is TRUE ?
- **(A)** Both Rule 1 and Rule 2 are L-attributed **(B)** Only Rule 1 is L-attributed
  (C) Only Rule 2 is L-attributed **(D)** Neither Rule 1 nor Rule 2 is L-attributed
- A → PQ
- P.i=A.i+2, P.i inherited from it's parent A.i .
- Q.i=P.i+A.i , Q.i inherited from it's parent A.i & left sibling P.i .
- A.s=P.s+Q.s ,A.s is synthesized from both childs.
- **Rule 1** is L-attributed
- A → XY
- X.i=A.i+Y.s , X.i inherited from it's parent A.i & right sibling Y.s . [not L-attributed]
- Y.i=X.s+A.i , Y.i inherited from it's parent A.i & left sibling X.s
- **Rule 2** is not L-attributed
- Ans: **(B)** Only Rule 1 is L-attributed

- GATE CS 2021,Set-1,Q4: Consider the following statements.
- S_1: The sequence of procedure calls corresponds to a preorder traversal of the activation tree.
- S_2: The sequence of procedure returns corresponds to a postorder traversal of the activation tree.
- Which one of the following options is correct?
- (A) S_1 is true and S_2 is false (B) S_1 is false and S_2 is true
- (C) S_1 is true and S_2 is true (C) S_1 is false and S_2 is false
- We can represent the activations of procedures during the running of an entire program by a tree, called an *activation tree*.
- S_1: activation tree preorder traversal is same as sequence of procedure calls ,True.
- S_2: activation tree postorder traversal is same as sequence of procedure returns ,True.
- Ans : (C) S_1 is true and S_2 is true

- GATE CS 2021,Set-1,Q26:Consider the following grammar (that admits a series of declarations, followed by expressions) and the associated syntax directed translation (SDT) actions, given as pseudo-code:
- P→ D*E*
- D→ int ID{record that ID.lexeme is of type int}
- D→ bool ID{record that ID.lexeme is of type bool}
- E→ E_1+E_2{check that E_1.type=E_2.type=int;set E.type :=int}
- E→ !E_1{check that E_1.type=bool; set E.type:=bool}
- E→ ID{set E.type :=int}
- With respect to the above grammar, which one of the following choices is correct?
- (A) The actions can be used to correctly type-check any syntactically correct program
- (B) The actions can be used to type-check syntactically correct integer variable declarations and integer expressions
- (C) The actions can be used to type-check syntactically correct boolean variable declarations and boolean expressions.
- (D) The actions will lead to an infinite loop
- Ans :(B) The actions can be used to type-check syntactically correct integer variable declarations and integer expressions

- GATE CS 2021,Set-1,Q50: Consider the following C code segment:
- a = b + c;
- e = a + 1;
- d = b + c;
- f = d + 1;
- g = e + f;
- In a compiler, this code segment is represented internally as a directed acyclic graph (DAG). The number of nodes in the DAG is ______________



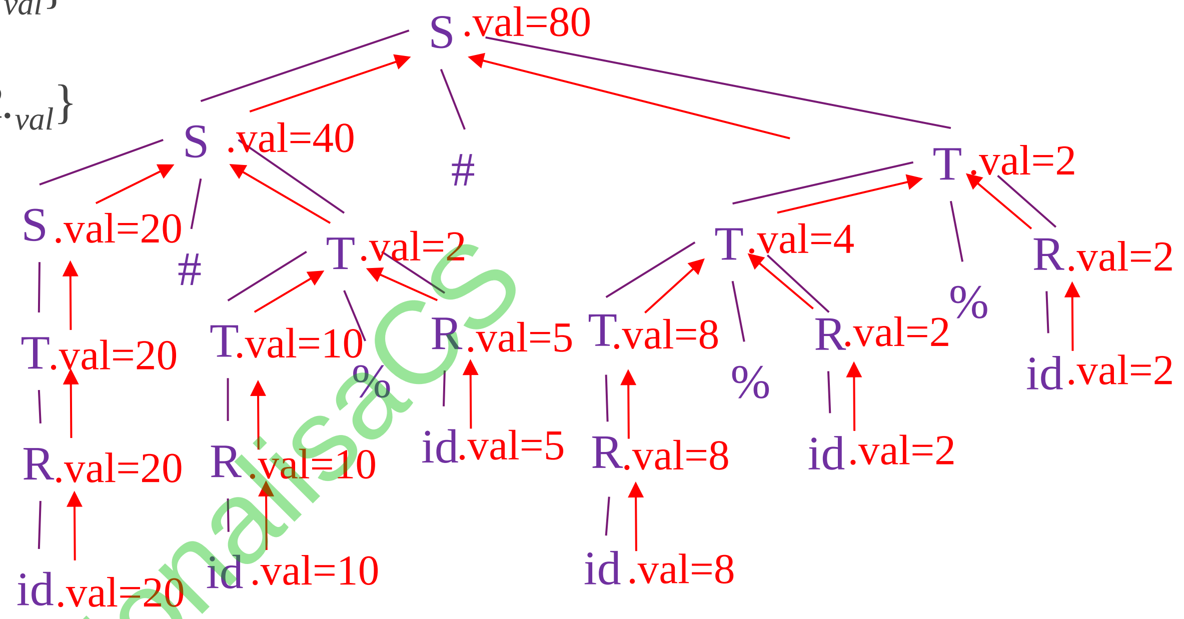
- Number of nodes=6
- Ans: 6

- GATE CS 2021,Set-2,Q13:In the context of compilers, which of the following is/are NOT an intermediate representation of the source program?
- (A) Three address code
- (B) Abstract Syntax Tree (AST)
- (C) Control Flow Graph (CFG)
- (D) Symbol table
- Intermediate code can be represented in following way:
- Non linear: Syntax Tree ,DAG ,Control Flow Graph
- Linear :Postfix code, Three-address code , SSA code
- (A) Three address code is an intermediate representation.
- (B) Abstract Syntax Tree is an intermediate representation.
- (C) CFG is an intermediate representation & used for code optimization.
- (D) Symbol Table is the abstract data structure use by compiler to store all the information about identifiers used in the program.
- Every phases of compiler interact with symbol table.
- Ans: (D) Symbol table

- GATE CS 2021,Set-2,Q38: For a statement S in a program, in the context of liveness analysis, the following sets are defined:
- *USE(S)* : the set of variables used in S
- *IN(S)* : the set of variables that are live at the entry of S
- *OUT(S)* : the set of variables that are live at the exit of S
- Consider a basic block that consists of two statements, S_1 followed by S_2. Which one of the following statements is correct?
- (A) $OUT(S_1)=IN(S_2)$
- (B) $OUT\ (S_1)=IN\ (S_2) \cup USE\ (S_1)$
- (C) OUT (S_1)=IN $(S_2) \cup$ OUT (S_2)
- (D) OUT (S_1)=USE $(S_1) \cup$IN (S_2)
- $OUT[B] = \cup_{S\ a\ successor\ of\ B}\ IN[S\ ]$
- $IN[B] = use_B\ U\ (OUT[B] - def_B)$
- (A) True
- (B) false due to USE (S_1)
- (C) false due to OUT (S_2)
- (D) false due to USE (S_1)
- Ans : (A) OUT(S1)=IN(S2)

- **GATE CS 2022 | Question: 55**
- Consider the following grammar along with translation rules.
- $S \rightarrow S_1 \# T$ $\{S._{val} = S_{1}._{val} * T._{val}\}$
- $S \rightarrow T$ $\{S._{val} = T._{val}\}$
- $T \rightarrow T_1 \% R$ $\{T._{val} = T_{1}._{val} \div R._{val}\}$
- $T \rightarrow R$ $\{T._{val} = R._{val}\}$
- $R \rightarrow id$ $\{R._{val} = id._{val}\}$
- Here # and % are operators and *id* is a token that represents an integer and $id._{val}$ represents the corresponding integer value. The set of non-terminals is $\{S, T, R, P\}$ and a subscripted non-terminal indicates an instance of the non-terminal.
- Using this translation scheme, the computed value of $S._{val}$ for root of the parse tree for the expression 20#10%5#8%2%2 is _______.
- Left-recursive rules indicates left associativity
- # and % are left associative.% have higher precedence than #.
- Replace # with * and % with / : 20 * 10 / 5 * 8 / 2 / 2
- = 20 * 2 * 8 / 2 / 2= 20 * 2 * 4 / 2
- = 20 * 2 * 2= 40*2 =80
- Ans : 80

- $S \rightarrow S_1 \# T$ $\{S._{val} = S_{1}._{val} * T._{val}\}$
- $S \rightarrow T$ $\{S._{val} = T._{val}\}$
- $T \rightarrow T_1 \% R$ $\{T._{val} = T_{1}._{val} \div R._{val}\}$
- $T \rightarrow R$ $\{T._{val} = R._{val}\}$
- $R \rightarrow id$ $\{R._{val} = id._{val}\}$
- Expression: 20#10%5#8%2%2
- Ans : 80



https://monalisacs.com/

GATE CS 2023 | Question: 26

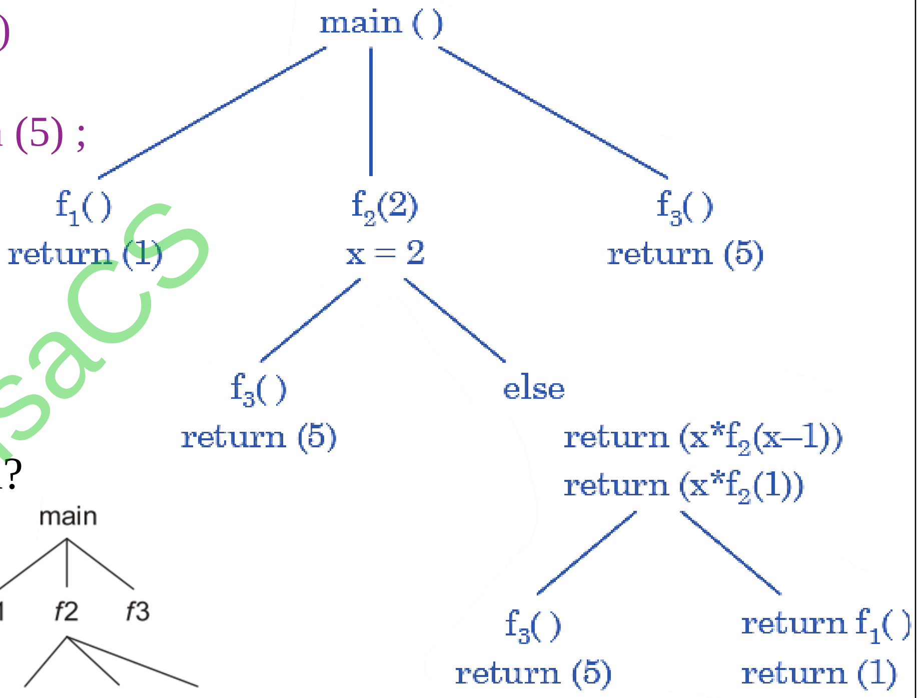
- Consider the following program :

```
int main()
{
    f1();
    f2(2);
    f3();
    return(0);
}
```

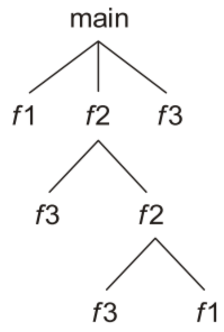
```
int f1( )
{
    return(1);
}
```

```
int f2 (int X)
{
    f3 ( ) ;
    if (X == 1)
        return f1 ( ) ;
    else
        return (X*f2(X-1));
}
```

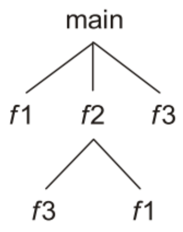
```
int f3 ( )
{
    return (5) ;
}
```



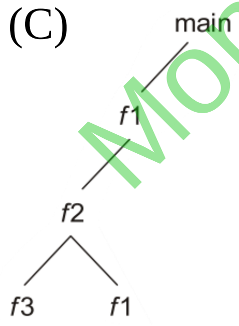
- Which one of the following options represents the activation tree corresponding to the main function?
- (A)



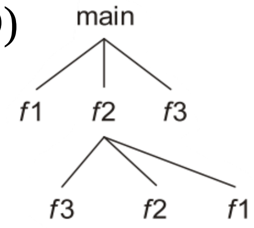
(B)



(C)



(D)

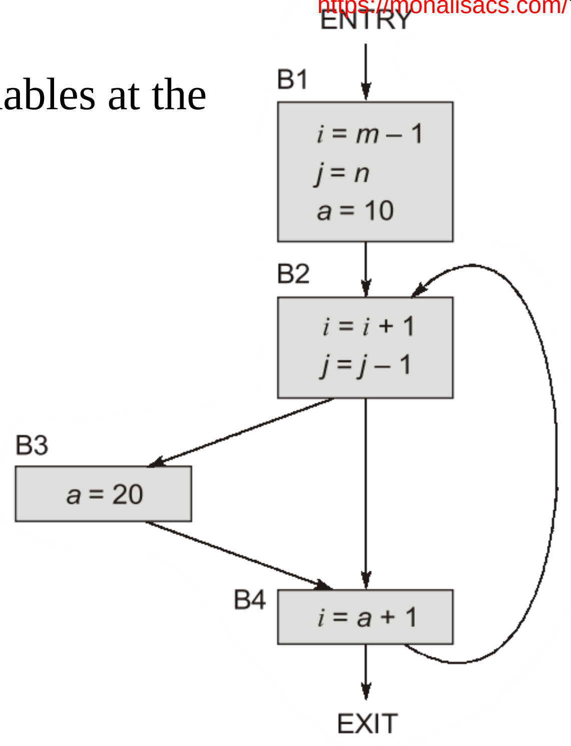


- Ans : (A)

https://www.youtube.com/@MonalisaCS

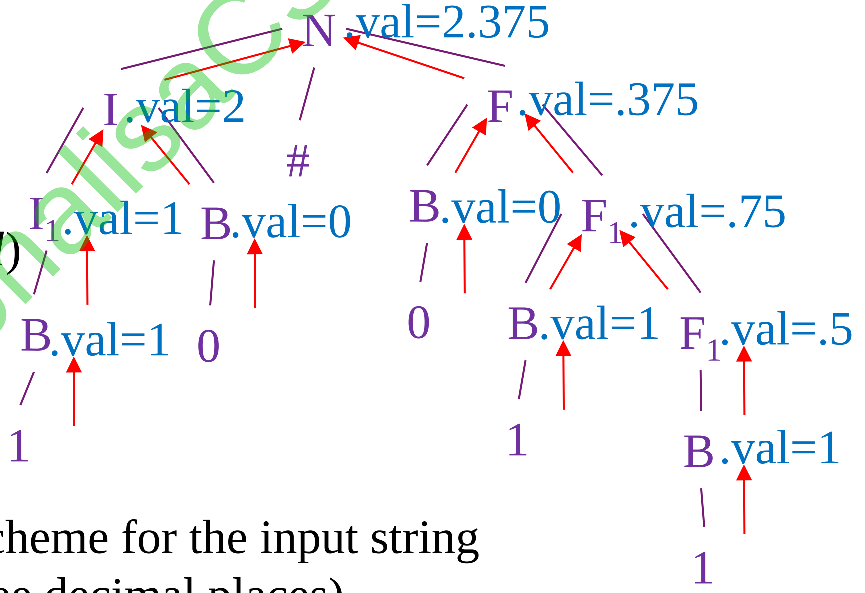
GATE CS 2023 | Question: 27

- Consider the control flow graph shown.
- Which one of the following choices correctly lists the set of *live* variables at the exit point of each basic block?
- (a) B1: { }, B2: {a}, B3: {a}, B4: {a}
- (b) B1: {i, j}, B2: {a}, B3: {a}, B4: {i}
- (c) B1: {a, i, j}, B2: {a, i, j}, B3: {a, i}, B4: {a}
- (d) B1: {a, i, j}, B2: {a, j}, B3: {a, j}, B4: {a, i, j}
- *OUT[B] =*$\cup_{S \text{ a successor of } B}$ *IN[S]*
- *IN[B] = use*$_B$ *U (OUT[B] - def*$_B$*) , IN[exit] = ∅*
- use_{B_1} ={m,n}, def_{B_1} ={i,j,a} ,use_{B_2} ={i,j}, def_{B_2} ={i,j}
- use_{B_3} ={ }, def_{B_3} ={a} ,use_{B_4} ={a}, def_{B_4} ={i}
- *OUT[B$_4$]=IN[exit] U IN[B$_2$] =∅ U {a,i,j}=***{a,i,j}**
- *IN[B$_4$] = {a} U ({a,i,j}- {i})={a,j}*
- *OUT[B$_3$]= IN[B$_4$] =***{a,j}**
- *IN[B$_3$] ={} U ({a,j}- {a})={j}*
- *OUT[B$_2$]= IN[B$_3$] U IN[B$_4$] ={j} U{a,j}=***{a,j}**
- *IN[B$_2$]={i,j} U({a,j,}-{i,j})={a,i,j}*
- *OUT[B$_1$]= IN[B$_2$]=***{a,i,j}**
- *IN[B$_1$]={m,n} U ({a,i,j}-{i,j,a})={m,n}*
- *OUT* are set of live variables at the exit point of each basic block.
- Ans : (D)



- **GATE CS 2023 | Question: 50**
- Consider the syntax directed translation given by the following grammar and semantic rules. Here N, I, F and B are non-terminals. N is the starting non-terminal, and #, 0 and 1 are lexical tokens corresponding to input letters "#", "0" and "1", respectively. X.*val* denotes the synthesized attribute (a numeric value) associated with a non-terminal X. I_1 and F_1 denote occurrences of I and F on the right hand side of a production, respectively. For the tokens 0 and 1, 0.*val* = 0 and 1.*val* = 1.

$N \rightarrow I \# F$	N.val = I.val + F.val
$I \rightarrow I_1 B$	I.*val* =(2I_1.*val*)+B.*val*
$I \rightarrow B$	I.*val* = B.*val*
$F \rightarrow BF_1$	F.*val* = ½ (B.*val*+F_1.*val*)
$F \rightarrow B$	F.*val* = ½ B.*val*
$B \rightarrow 0$	B.*val* = 0.*val*
$B \rightarrow 1$	B.*val* = 1.*val*



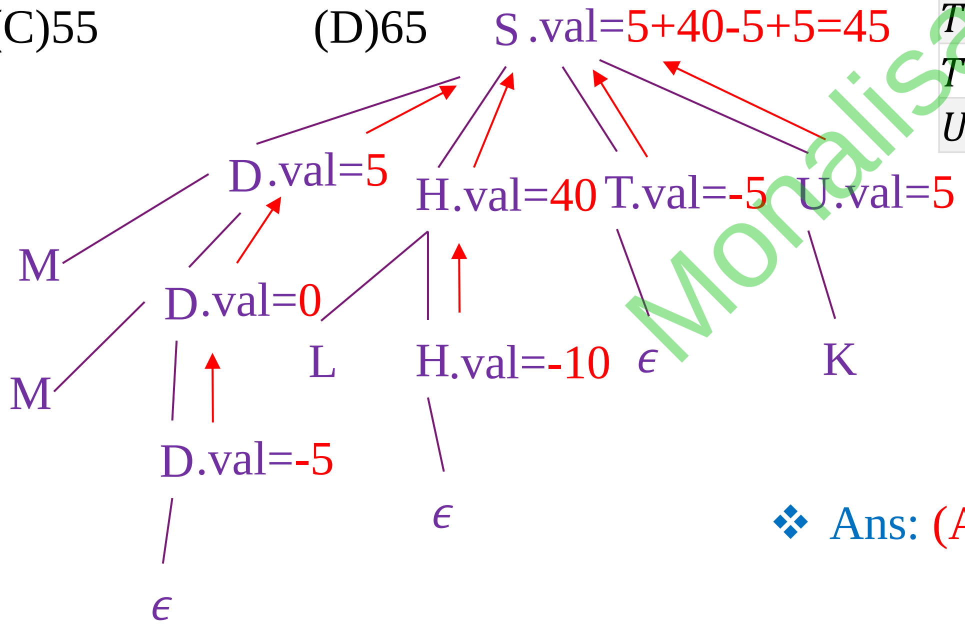
- The value computed by the translation scheme for the input string 10#011 is__2.375___. (Rounded off to three decimal places)

- **GATE CS 2024 | Set 2 | Question: 19**
- Which of the following statements is/are FALSE?
- (A)An attribute grammar is a syntax-directed definition (SDD) in which the functions in the semantic rules have no side effects
- (B)The attributes in a L-attributed definition cannot always be evaluated in gumv yua depth-first order
- (C)Synthesized attributes can be evaluated by a bottom-up parser as the input is parsed
- (D)All L-attributed definitions based on LR(1) grammar can be evaluated using a bottom-up parsing strategy
- (A) A syntax-directed definition (SDD) is a context-free grammar with attributes and semantic rules to evaluate the attributes
- Attribute grammars are SDDs with no side effects ,True
- (B) L-attributed definitions are a class of syntax-directed definitions where attributes can always be evaluated in depth first order , False
- (C) Synthesized Attributes can be evaluated by a bottom-up parser, True
- (D) Attributes are evaluated top down ,pre order traversal or depth first traversal , False
- Ans: B,D

GATE CS 2024 | Set 1 | Question: 27

- Consider the following syntax-directed definition (SDD).
- Given "MMLK" as the input, which one of the following options is the CORRECT value computed by the SDD (in the attribute $S.val$)?
- (A)45 (B)50
- (C)55 (D)65

$S \rightarrow DHTU$	$\{S.val = D.val + H.val + T.val + U.val\};$
$D \rightarrow "M" D_1$	$\{D.val = 5 + D_1.val\};$
$D \rightarrow \epsilon$	$\{D.val = -5\};$
$H \rightarrow "L" H_1$	$\{H.val = 5 * 10 + H_1.val\};$
$H \rightarrow \epsilon$	$\{H.val = -10\};$
$T \rightarrow "C"\ T_1$	$\{T.val = 5 * 100 + T_1.val;\};$
$T \rightarrow \epsilon$	$\{T.val = -5\};$
$U \rightarrow "K"$	$\{U.val = 5\};$



❖ Ans: (A)45

GATE CS 2024 | Set 2 | Question: 33

- Consider the following expression: $x[i]=(p+r)*-s[i]+u/w$. The following sequence shows the list of triples representing the given expression, with entries missing for triples (1),(3), and (6).
- Which one of the following options fills in the missing entries CORRECTLY?
- (A) (1) =[]si (3)$*$(0)(2) (6) []=xi
- (B) (1)[]=si (3)$-$(0)(2) (6)=[]x(5)
- (C) (1)=[] si (3)$*$(0)(2) (6)[]=x(5)
- (D) (1)[] =si (3)-(0)(2) (6)=[] xi
- Ans :(A) (1) =[]si (3)$*$(0)(2) (6) []=xi

(0)	+	p	r
(1)	= []	si	
(2)	uminus	(1)	
(3)	$*$	0	2
(4)	/	u	w
(5)	+	(3)	(4)
(6)	[]=	xi	
(7)	=	(6)	(5)

GATE CS 2024 | Set 1 | Question: 29

- Consider the following pseudo-code.

```
L1: t1 = -1                      B_1
L2: t2 = 0
L3: t3 = 0                       B_2
L4: t4 = 4 * t3                  B_3
L5: t5 = 4 * t2
L6: t6 = t5 * M
L7: t7 = t4 + t6
L8: t8 = a[t7]
L9: if t8 <= max goto L11
L10: t1 = t8                     B_4
L11: t3 = t3 + 1                 B_5
L12: if t3 < M goto L4
L13: t2 = t2 + 1                 B_6
L14: if t2 < N goto L3
L15: max = t1                    B_7
```

- Which one of the following options CORRECTLY specifies the number of basic blocks and the number of instructions in the largest basic block, respectively?
- (A) 6 and 6 (B) 6 and 7 (C) 7 and 7 (D) 7 and 6
- The rules for finding leaders are:

1. The first three-address instruction in the intermediate code is a leader.
2. Any instruction that is the target of a conditional or unconditional jump is a leader.
3. Any instruction that immediately follows a conditional or unconditional jump is a leader.

- Leader: 1,3,4,10,11,13,15
- Number of basic blocks=7
- Number of instructions in the largest basic block=6
- Ans : (D)7 and 6