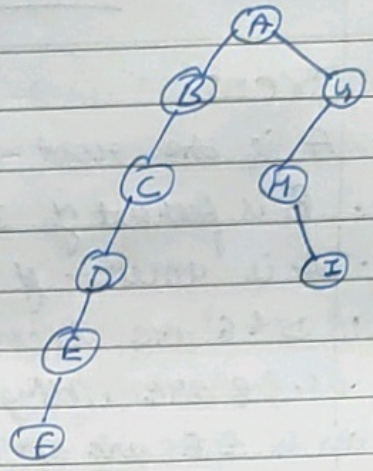
Page No.
Date

Homework 6.1



1. A is the root of the tree : True
2. f is the root of the tree : False
3. The leaves of the tree are : f and I
4. The ancestors of I are : H, G and A
5. H is a sibling of c : False
6. The height of the tree is : 5
7. The children of G are : H
8. It is a binary tree : True
9. Degrees of A, C and f are : A: 2 , C: 1 , f: 0
10. The root of right subtree of A is : G

Page No. _____
Date

Homework 6.2

Note the following points when arithmetic expression is expressed in form of binary trees.

- All the numerical values are present at the leaves.
- None of the node has exactly one child. Having exactly one child is meaningless when we have operators that require 2 operands.

Note: If the enum OperatorType had operators like factorial, square-root etc. that require single operand, node can have exactly one child.

```
Algorithm solve (Node node)
  if (node.type == NUM)
        return node.number
  switch (node.operation)
  case PLUS
     return solve(node.left) + solve(node.right)
  case MINUS
     return solve(node.left) - solve(node.right)
  case MULT
     return solve(node.left) * solve(node.right)
  case DIV
     return solve(node.left) / solve(node.right)
```

Homework 6.3

To make a binary tree having minimum nodes of given height, we will make a tree such that every node contributes to increment in height.

total nodes = h+1

- This tree somewhat resembles singly linked list.

return node.parent

Homework 6.5

```
public boolean isRoot(Node node){
    return node.parent==NULL;
}

public boolean isLeaf(Node node){
  return node.left==NULL && node.right==NULL;
}

public boolean isInternal(Node node){
  return ! isLeaf(node);
```

Homework 6.4

(a) Distinct binary trees of height n-1 with n nodes:
This is one of the extreme case when every node except leaf node has exactly one child.
Putting node on left and right would generate a new binary tree.
1 node is root node. Remaining n-1 nodes have 2 choices each.
Total possible trees: 2 x 2 -- n-1 times

$= 2^{n-1}$.

Page No.
Date

Ans

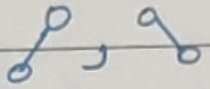
(b) Number of possible binary trees with n nodes =

1. 1 if n = 1
2. n × (number of possible Binary trees with n-1 nodes) if n ≠ 1

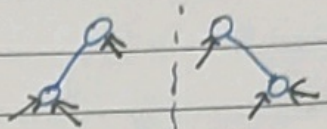
Verification

n = 1

n = 2



n = 3

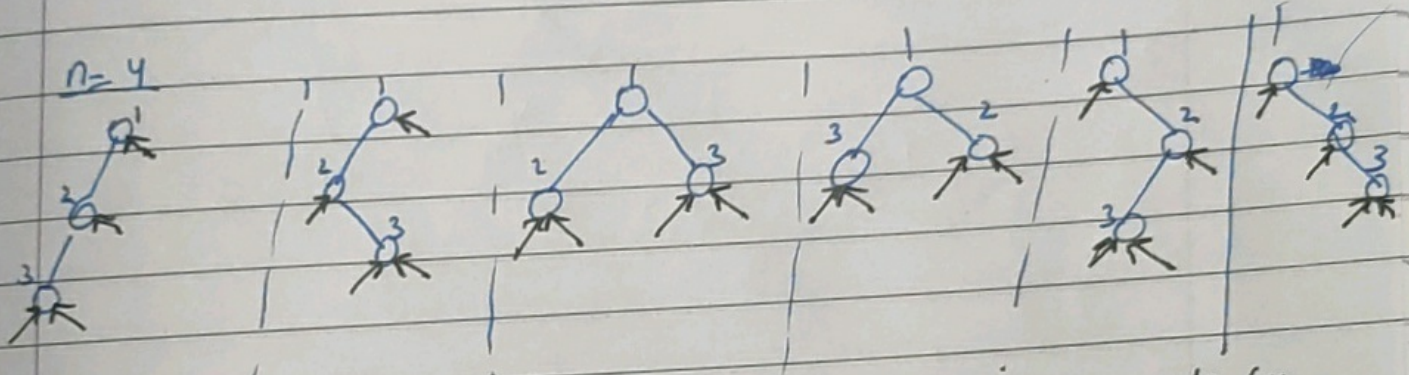


3 vacant pos.

3 vacant pos

Ans = 6.

n = 4



Total 4 possible trees for each case made by n = 3.