**자료구조론 CC343\_2207**

**Reading assignment 9**

**경기대학교 컴퓨터공학부**

**201511837 이상민**

**Review Questions**

1. Explain the concept of a tree. Discuss its applications.  
tree의 개념을 설명하라. 응용 프로그램에 대해 논의하십시오.

**트리는 노드로 이루어진 자료구조를 의미한다.**

1. **트리는 하나의 루트 노드를 갖는다.**
2. **루트 노드는 0개 이상의 자식 노드를 갖고 있다.**
3. **그 자식 노드 또한 0개 이상의 자식 노드를 갖고 있고, 이는 반복적으로 정의된다.**

**트리의 주요 용도**

**1. 계층 데이터를 조작합니다.**

**2. 정보를 쉽게 검색할 수 있도록 합니다.**

**3. 정렬된 데이터 목록을 조작합니다.**

**4. 시각 효과를 위해 디지털 이미지를 합성하는 워크 플로우로 사용합니다.**

**5. 라우터 알고리즘**

**6. 다단계 의사 결정 양식이다.**

2. What are the two ways of representing binary trees in the memory? Which one do you prefer and why?  
기억 속의 이진수를 나타내는 두 가지 방법은 무엇인가? 너는 어떤 것을 더 선호하고 왜 좋아하니?

**1. Linked Representation of Binary Tree**

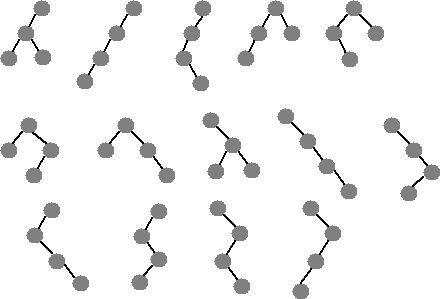
**2. Sequential representation of Binary Tree**

**In array representation of binary tree we use one dimensional array(1-D array).**

**While in linked list representation of binary tree we use Linked List data structure.**

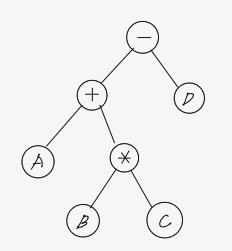
**Array representation is best idea when the binary tree is Almost Complete Binary Tree or simply Complete Binary Tree. Otherwise Link list representation is the best idea.**

3. List all possible non-similar binary trees having four nodes.  
4개의 노드를 가진 유사하지 않은 이진 트리를 모두 나열한다.



4. Draw the binary expression tree that represents the following postfix expression:  
다음 수정 후 식을 나타내는 이진 식 트리를 그린다.

A B + C \* D –



5. Write short notes on:

(a) Complete binary trees (완전 이진 트리)

**레벨 k에 있는 최대 노드의 수는 2^(k-1)이다. (단, k = 1, 2, 3 ...)**

**높이가 h인 포화이진트리에 있는 노드의 수는 2^h-1이다.**

**완전이진트리는 마지막 레벨을 제외한 나머지가 모두 2개의 노드 쌍으로 이루어져 있기 때문에 리스트 중간이 비어 있을 수가 없다.**

(b) Extended binary trees (확장 이진 트리)  
**Extended binary tree is a type of binary tree in which all the null sub tree of the original tree are replaced with special nodes called external nodes whereas other nodes are called internal nodes**

(c) Tournament trees (토너먼트 트리)

**Tournament tree is a complete binary tree n external nodes and n-1 internal nodes. The external nodes represent the players and internal nodes represent the winner of the match between the two players.**

(d) Expression trees

**A binary expression tree is a specific kind of a used to represent expressions. Two common types of expressions that a binary expression tree can represent are and . These trees can represent expressions that contain both and operators.**

(e) Huffman trees

**Huffman coding is a lossless data compression algorithm. In this algorithm, a variable-length code is assigned to input different characters. The code length is related to how frequently characters are used. Most frequent characters have the smallest codes and longer codes for least frequent characters.**

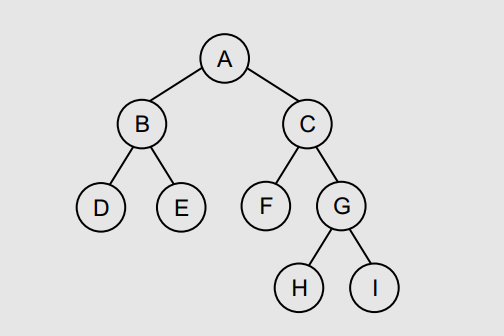
(f) General trees

**A general tree is a tree where each node may have zero or more children**

(g) Forests

**Forest is a collection of disjoint trees. In other words, we can also say that forest is a collection of an acyclic graph which is not connected.**

6. Consider the tree given below. Now, do the following:



(a) Name the leaf nodes **: D, E, F, H, I**

(b) Name the non-leaf nodes **: A, B, C, G**

(c) Name the ancestors of E **: A, B**

(d) Name the descendants of A **: B, C, D, E, F, G, H, I**

(e) Name the siblings of C **: B**

(f) Find the height of the tree **: 3**

(g) Find the height of sub-tree rooted at E **: 2**

(h) Find the level of node E **: 2**

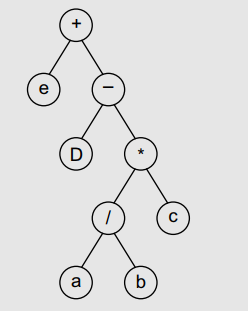
(i) Find the in-order, pre-order, post-order, and level-order traversal

**in-order : D-B-E-A-F-C-H-G-I**

**pre-order : A-B-D-E-C-F-G-H-I**

**post-order : D-E-B-H-I-G-F-C-A**

7. For the expression tree given below, do the following:



(a) Extract the infix expression it represents

**Infix : (E + (D - ((a / b) \* c )))**

(b) Find the corresponding prefix and postfix expressions

**Prefix : ( E ( D ( C (ab/) \* ) - ) + )**

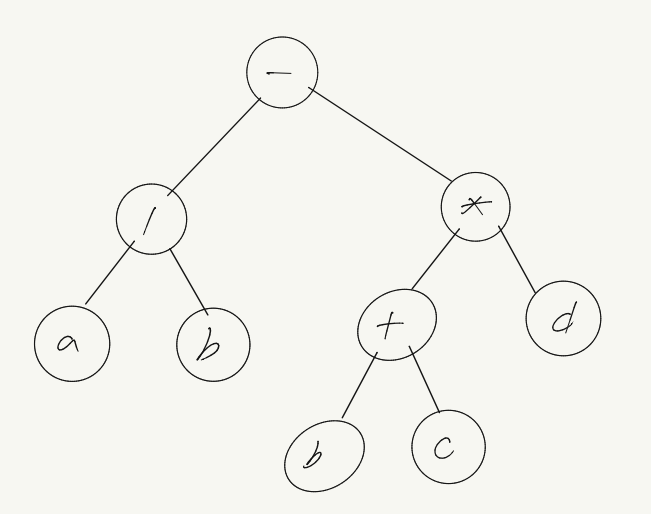
**Postfix : ( +E ( -D ( \*C ( /ab ) ) ) )**

(c) Evaluate the infix expression, given a = 30, b = 10, c = 2, d = 30, e = 10

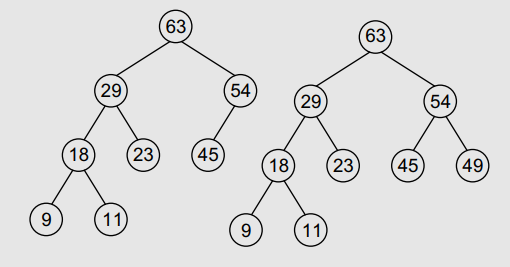
**= 34**

8. Convert the prefix expression –/ab\*+bcd into infix expression and then draw the corresponding expression tree.

**Prefix to infix : ( a / b ) – ( ( b + c ) \* d )**



9. Consider the trees given below and state whether they are complete binary tree or full binary tree.  
아래에 제공된 트리를 고려하여 완전한 이진 트리인지 아니면 완전한 이진 트리인지 명시하십시오.



**Answer : complete binary tree**

10. What is the maximum number of levels that a binary search tree with 100 nodes can have?  
100개의 노드를 가진 이진 검색 트리가 가질 수 있는 최대 레벨 수는?

**Answer : 99**

11. What is the maximum height of a tree with 32 nodes?  
32개의 노드가 있는 나무의 최대 높이는 얼마인가?

**Answer : 32**

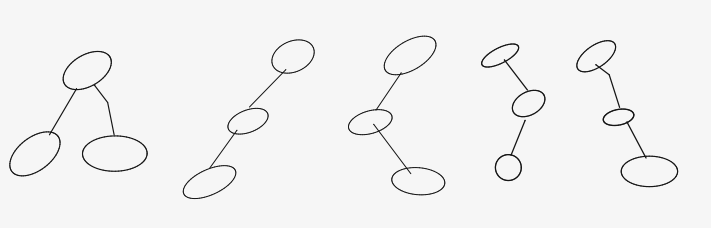
12. What is the maximum number of nodes that can be found in a binary tree at levels 3, 4, and 12? 레벨 3, 4, 12의 이진 트리에서 찾을 수 있는 최대 노드 수는 얼마인가?

**레벨3 = = 8**

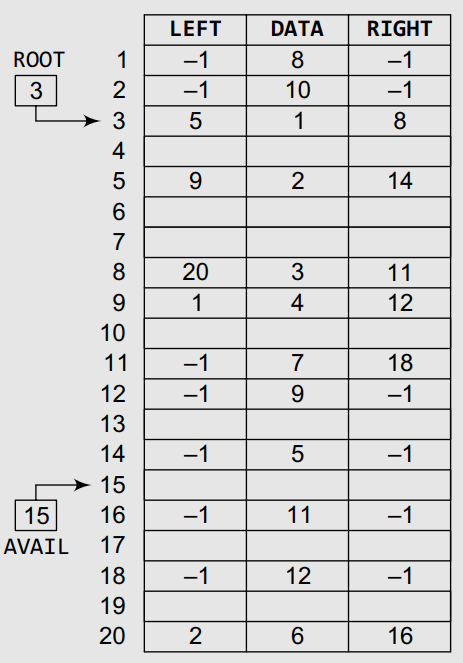
**레벨4 =**

**레벨12 = = 4096**

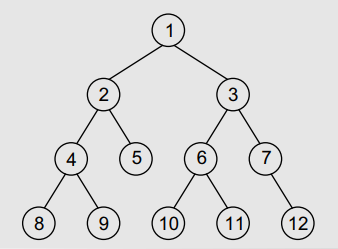
13. Draw all possible non-similar binary trees having three nodes.  
3개의 노드가 있는 유사하지 않은 이진수를 모두 그린다.

****

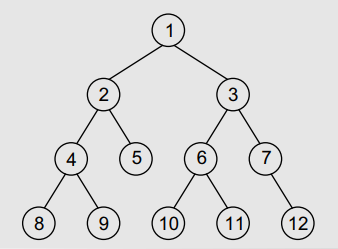
14. Draw the binary tree having the following memory representation:  
다음과 같은 메모리 표현을 가진 이진 트리를 그린다.



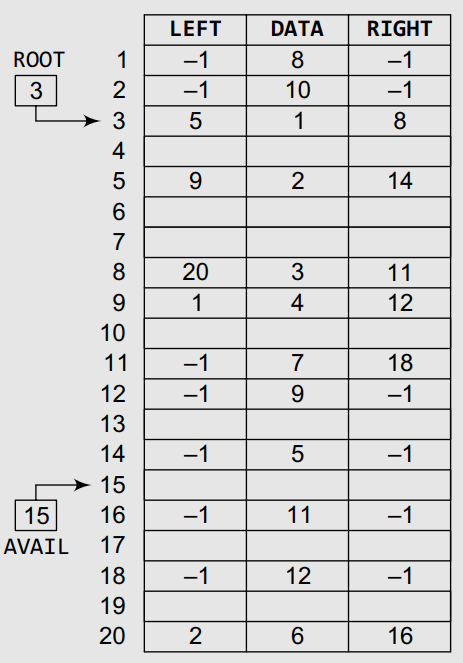
**Answer :**



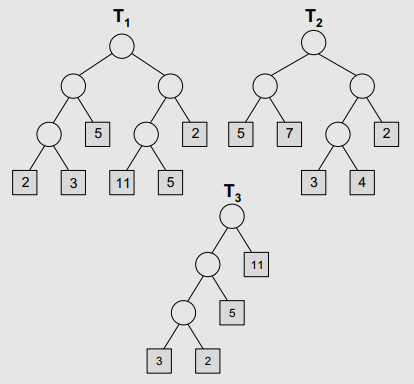
15. Draw the memory representation of the binary tree given below.



**Answer :**



16. Consider the trees T1, T2 and T3 given below and calculate their weighted path lengths.  
아래에 제시된 T1, T2 및 T3 나무를 고려하여 가중 경로 길이를 계산하십시오.

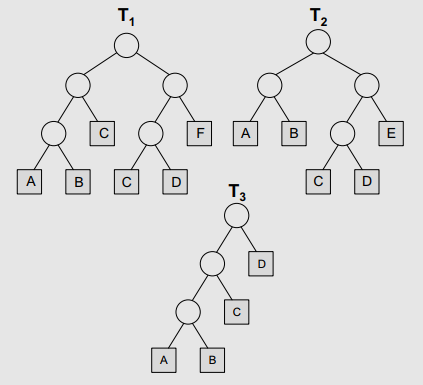


**T1 : 2\*3 + 3\*3 + 5\*2 + 11\*3 + 5\*3 + 2\*2 = 77**

**T2 : 5\*2 + 7\*2 + 3\*3 + 4\*3 + 2\*2 = 49**

**T3 : 3\*3 + 2\*3 + 5\*2 + 11\*1 = 36**

17. Consider the trees T1, T2, and T3 given below and find the Huffman coding for the characters.  
아래에 제시된 T1, T2, T3 트리를 고려하여 캐릭터에 대한 Huffman 코딩을 찾으십시오.



**T1 : A = 000, B = 001, C = 01, 100 D = 101, F = 11**

**T2 : A = 00, B = 01, C = 100, D = 101, E = 11**

**T3 : A = 000, B = 001, C = 01, D = 1**

**multiple-choice Questions**

1. Degree of a leaf node is \_\_\_\_\_\_.

**(a) 0** (b) 1 (c) 2 (d) 3

2. The depth of root node is \_\_\_\_\_\_.

**(a) 0** (b) 1 (c) 2 (d) 3

3. A binary tree of height h has at least h nodes and at most \_\_\_\_\_\_ nodes.

(a) 2h (b) (c) **(d) -1**

4. Pre-order traversal is also called \_\_\_\_\_\_.

**(a) Depth first**

(b) Breadth first

(c) Level order

(d) In-order

5. The Huffman algorithm can be implemented using a \_\_\_\_\_\_.

(a) Dequeue

(b) Queue

**(c) Priority queue**

(d) None of these

6. Total number of nodes at the nth level of a binary tree can be given as

(a)

(b)

(c)

**(d)**

**True or False**

1. Nodes that branch into child nodes are called parent nodes. **: True**

2. The size of a tree is equal to the total number of nodes. **: True**

3. A leaf node does not branch out further. **: True**

4. A node that has no successors is called the root node. **: False**

5. A binary tree of n nodes has exactly n – 1 edges. **: True**

6. Every node has a parent. **: False**

7. The Huffman coding algorithm uses a variable- length code table. **: True**

8. The internal path length of a binary tree is defined as the sum of all path lengths summed over each path from the root to an external node. **: False**

**Fill in the Blanks**

1. Parent node is also known as the \_\_\_\_\_\_ node.  
**answer : Ascendant**

2. Size of a tree is basically the number of \_\_\_\_\_\_ in the tree.  
**answer : Nodes**

3. The maximum number of nodes at the kth level of a binary tree is \_\_\_\_\_\_.  
**answer : -1**

4. In a binary tree, every node can have a maximum of \_\_\_\_\_\_ successors.  
**answer : Two**

5. Nodes at the same level that share the same parent are called \_\_\_\_\_\_.  
**answer : Siblings**

6. Two binary trees are said to be copies if they have similar \_\_\_\_\_\_ and \_\_\_\_\_\_.  
**answer : Structure and contents**

7. The height of a binary tree with n nodes is at least \_\_\_\_\_\_ and at most \_\_\_\_\_\_.  
**answer : n and**

8. A binary tree T is said to be an extended binary tree if \_\_\_\_\_\_.  
**answer : Each node in the tree has either no child or exactly two children**

9. \_\_\_\_\_\_ traversal algorithm is used to extract a prefix notation from an expression tree.  
**answer : Preorder**

10. In a Huffman tree, the code of a character depends on \_\_\_\_\_\_.  
**answer : The estimated probability of occurrence for each possible value of the source character.**