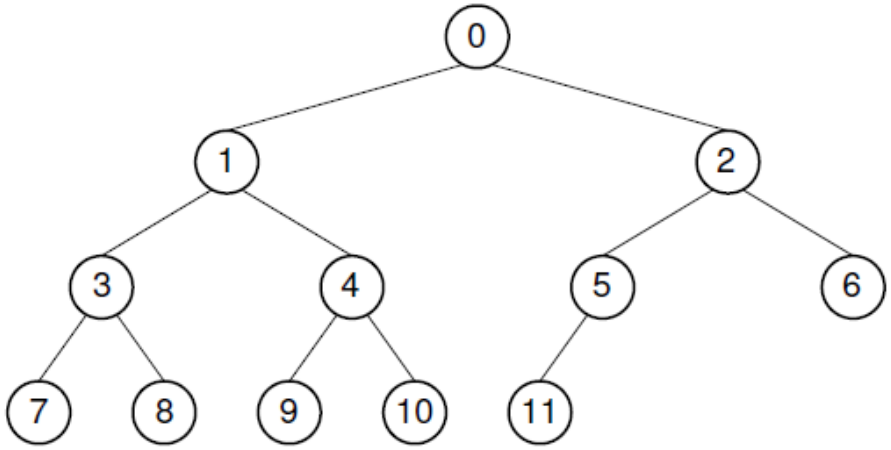
# DATA STRUCTURES AND ALGORITHMS

PRACTICE No.4

# Array implementation of the tree




(a)

| Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parent | – | 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 |
| Left Child | 1 | 3 | 5 | 7 | 9 | 11 | – | – | – | – | – | – |
| Right Child | 2 | 4 | 6 | 8 | 10 | – | – | – | – | – | – | – |
| Left Sibling | – | – | 1 | – | 3 | – | 5 | – | 7 | – | 9 | – |
| Right Sibling | – | 2 | – | 4 | – | 6 | – | 8 | – | 10 | – | – |

(b)

**Figure 5.12** A complete binary tree and its array implementation. (a) The complete binary tree with twelve nodes. Each node has been labeled with its position in the tree. (b) The positions for the relatives of each node. A dash indicates that the relative does not exist.

# Array implementation of the tree

The formulae for calculating the array indices of the various relatives of a node are as follows. The total number of nodes in the tree is $n$. The index of the node in question is $r$, which must fall in the range 0 to $n - 1$.

- Parent$(r) = \lfloor (r-1)/2 \rfloor$ if $r \neq 0$.
- Left child$(r) = 2r + 1$ if $2r + 1 < n$.
- Right child$(r) = 2r + 2$ if $2r + 2 < n$.
- Left sibling$(r) = r - 1$ if $r$ is even.
- Right sibling$(r) = r + 1$ if $r$ is odd and $r + 1 < n$.

# Exercises

**No.1** Write a C program to check if the particular value **K** exists in the binary tree. If yes, then check is **K** a leaf or an internal node. Print out the parent node and the child nodes (left, right). Use array implementation of the binary tree.

**No.2** Write a C program to built binary search tree (BST) based on array implementation.

**Homework**

- Write a C program to merge two singly linked lists.

# Homework

Write C program, that will draw diamond from **D** letters as follows. Dimensions of the diamond *n x n*, where *n* is odd number. Input data is an integer *n*.

```
*  *  *  *  D  *  *  *  *
*  *  *  D  D  D  *  *  *
*  *  D  D  D  D  D  *  *
*  D  D  D  D  D  D  D  *
D  D  D  D  D  D  D  D  D
*  D  D  D  D  D  D  D  *
*  *  D  D  D  D  D  *  *
*  *  *  D  D  D  *  *  *
*  *  *  *  D  *  *  *  *
```