National University
Of Computer and Emerging Sciences

CL2001 – Data Structure Lab

Exercise # 11

Note:

- Copied task will be awarded **zero** marks.
- Use comments wherever applicable.
- Note that these lab task marks could be graded through a viva in lab.
- Variables and functions names should be meaningful.

Problem: 1 | Check array is max heap or not

Given an array, write a program to check if the given array represents a Max Heap.

Input:

arr[] = {90, 15, 10, 7, 12, 2}

Output: True

The given array represents below tree

```
          90

        /        \

   15              10

 /    \              /

7      12        2
```

The tree follows max-heap property as every node is greater than all of

its descendants.

Input:

arr[] = {9, 15, 10, 7, 12, 11}

Output: False

The given array represents below tree

```
              9
          /         \
      15                10
    /     \             /
  7        12        11
```

The tree doesn't follow max-heap property 9 is smaller than 15 and 10, and 10 is smaller than 11.

Hint:

A Simple Solution is to first check root if it’s greater than all of its descendants. Then check for children of the root. Time complexity of this solution is O(n2)

OR

An Efficient Solution is to compare root only with its children (not all descendants), if root is greater than its children and the same is true for all nodes, then tree is max-heap (This conclusion is based on transitive property of > operator, i.e., if x > y and y > z, then x > z).

Problem: 2 | Build Min Heap

Given an array of N elements. The task is to build a Min Heap from the given array.

Input: *arr[] = {4, 10, 3, 5, 1}*
Output: *Corresponding Min-Heap:*

```
     1
   /  \
  4    3
 / \
5   10
```

Output: *arr[] = {1, 4, 3, 5, 10}*

Problem: 3 | Convert min Heap to max Heap

Given array representation of min Heap, convert it to max Heap in O(n) time.

Example:

Input: arr[] = [3 5 9 6 8 20 10 12 18 9]

```
                 3
              /      \
          5              9
        /   \          /   \
      6       8      20    10
    /   \    /
  12    18   9
```

Output: arr[] = [20 18 10 12 9 9 3 5 6 8]

OR any Max Heap formed from input elements

```
                 20
              /      \
          18             10
        /   \          /   \
      12      9      9      3
    /   \    /
   5     6   8
```

The problem might look complex at first look. But our final goal is to only build the max heap. The idea is very simple – we simply build Max Heap without caring about the input. We start from bottom-most and rightmost internal node of min Heap and heapify all internal modes in bottom up way to build the Max heap.