Practice 4 : Heaps

1. For the given data as integers perform following operations in min-heap:
   a. Create heap
   b. Display heap elements in sequence after k deletions of root element
   Input: (T, n, k, {x_i})
   2
   11
   4
   12 1 21 2 24 23 15 26 4 33 10
   7
   2
   5 2 8 1 4 6 10
   Output:
   1 2 15 4 10 23 21 26 12 33 24
   12 24 15 26 33 23 21
   1 2 6 5 4 8 10
   4 5 6 8 10

2. Implement max-heap and perform insertion and deletion Operations.
   a. Insert the element
   b. Delete the element
   c. Display all elements
   d. Quit
   Input: (n, x_i)
   a 30
   a 50
   a 70
   b 10
   b 50
   a 100
   c
   d
   Output:
   Inserted
   Inserted
   Inserted
   10 not found
   Deleted
   100 30 70

3. Implement heapsort.
   Input: (T, n_i, {x_i})

2
6
12 11 20 5 16 7
8
15 24 16 22 5 20 40 8
Output:
5 7 11 12 16 20
5 8 15 16 20 22 24 40

4. Implement binary heap using a binary tree (not arrays). Binary heap operations are:
   1. Insert
   2. Delete min
   3. Check full
   4. Check empty
   5. Quit

   Input: (T, n, k, {x_i})
   10
   1 24
   1 6
   1 28
   1 5
   1 63
   1 19
   1 94
   2
   2
   2
   2
   2
   2
   2
   4
   5
   Output:
   24
   6 24
   6 24 28
   5 6 24 28
   5 6 28 24 63
   5 6 19 24 63 28
   5 6 19 24 63 28 94
   6 24 19 94 63 28
   19 24 28 94 63
   24 63 28 94

28 63 94
63 94
94
Empty
underflow
True

5. For a given array of elements, determine the minimum number of interchanges needed to convert it into a max-heap.
   Input: (T, n, {x_i})
   2
   13
   89 19 50 17 12 15 2 5 7 11 6 9 100
   8
   15 24 16 22 5 20 40 8
   Output:
   3
   5

6. Write a program to construct priority queue using heap. Print the final contents of the priority queue.
   Input: (n, {x_i})
   8
   15 24 32 2 5 28 48 16
   Output:
   48 16 32 15 5 24 28 2

Practice 5 : Binary trees

1. Read n ints and make a binary search tree (BST). Do k search operations to print results as y/n.
   Input: (n, x_i, k, y_i)
   4
   2 1 4 3
   3
   3 7 1
   Output:
   y
   n
   y

2. Read n ints and make a BST in the same order. Print the tree in preorder, inorder and postorder traversals. Separate characters by '_'.
   Input: (n, x_i)
   4
   2 1 4 3
   Output:
   2_1_4_3_
   1_2_3_4_
   1_3_4_2_

3. Read 2n ints. Use each half to create two BSTs in the given order. Find if the two trees are identical. Print y/n. There are T test cases.
   Input: (T, n, x_i)
   3
   3
   1 2 3 1 3 2
   1 2 3 2 3 1
   2 1 3 2 3 1
   Output:
   n
   n
   y

4. Given a BST, print out all root-to-leaf paths.
5. Find the number of leaves in a BST.
6. Find sum of all the leaf nodes in a BST.
7. Construct a binary tree given inorder and post-order traversal outputs.
8. Construct a full binary tree from given pre-order and post-order traversals and print in-order traversal of it.
9. Delete a BST. Print the order in which nodes are deleted.
10. Construct the mirror tree of a given BST.
11. Find out the in-order successor and predecessor of a given node in a BST.
12. Given a BST and a key, write a function that prints all the ancestors of the key in the given binary tree.
13. Write a function which deletes all the terminal nodes in BST.
14. A SumTree is a Binary Tree where the value of a node is equal to the sum of the nodes present in its left subtree and right subtree. Write a function that returns 1 if the given BST is SumTree and 0 otherwise. All leaf nodes are trivial SumTrees.
15. Write a function to print all the nodes in a BST along with their individual heights and depths.
16. Print output of depth-first search given a BST.
17. Print output of breadth-first search given a BST.
18. Delete all duplicates of a node from a given binary search tree.