IIT PALAKKAD

COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology, Palakkad
CS2130: Data Structures and Algorithms Lab
Lab 9 (Binary Trees)

5 Oct, 2018

Time: 3 hrs

General Instructions

- You are expected to follow good programming practices (refer to the book titled *C How to Program* by Dietel and Dietel). Your programs should handle all kinds of test cases - positive, negative, corner cases/boundary conditions, good input, bad input and are never expected to crash on any input.
- You are also expected to provided a high level description of the functions used in your programs. There is no restriction on the format except that it should be concise and precise. Further, the running time and additional space (excluding the space for the arguments) used by the functions should be clearly stated and justified.
- **Please note that poor programming practices and improper documentation can attract a penalty of up to 20% of the marks allocated to a question.**

1. (30 points) Given an undirected graph G, a distinguished vertex $s \in V(G)$ and a non-negative integer k, write a program to find the set of vertices that are at a distance at most k from s.

   Input Format: The input is an undirected graph G on n vertices and m edges where $n \geq 1$. The vertex set of G is $\{0, 1, \ldots, n-1\}$ and an edge between vertex i and vertex j is denoted by (i, j) or (j, i). The input file will contain $m+3$ lines with the first line containing the number n. The second line contains the vertex s and the third line contains the value of k. Each of the subsequent m lines will contain two non-negative integers i and j, separated by a space, denoting edge (i, j) of G.

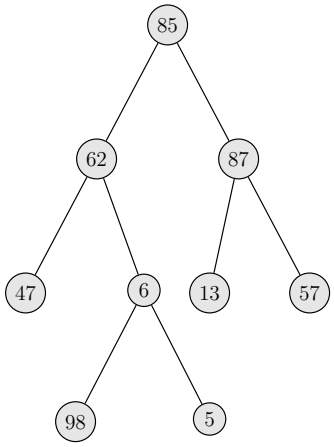
   Output Format: The output consists of a single line containing the vertices that are at a distance at most k from s in G. Two consecutive vertices are to be separated by a space and the vertices are to be displayed in the increasing order of their identifiers.

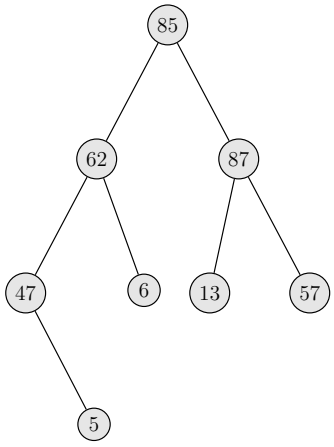
```
Sample Input:
10
1
4
0 1
1 2
2 3
3 4
4 5
5 6
```

Continued on next page...

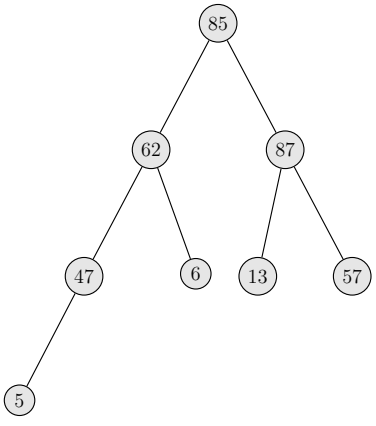
```
7 8
8 9

Expected Output:
0 1 2 3 4 5
```

2. (40 points) Write a program to build a nearly complete binary tree (with positive integer values in every vertex) and print its preorder, inorder and postorder traversal sequences. Recall that a nearly complete binary tree is a binary tree that is completely filled on all levels except possibly the lowest, which is filled from the left up to a point. For example, the following two binary trees are not nearly complete binary trees.





However, the following binary tree is a nearly complete binary tree.

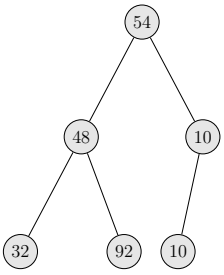


Input Format: The input consists of $n+1$ lines where the first line contains the number n of vertices in the tree T. The subsequent n lines contain the values at the vertices of T, one value per line. You should use the approach described in the earlier lectures to create the tree using the values provided. That is, the values have to be inserted in the same order as mentioned in the input, filling each level one after the other from left to right. For example, consider the following example input.

Continued on next page...

```
Sample Input:
6
54
48
10
32
92
10
```

The tree T corresponding to this input is shown below.



Output Format: The output consists of three lines containing the inorder, preorder and postorder traversal sequences of T, one sequence per line. In each of these lines, two consecutive vertices are separated by a space.

```
Sample Input:
6
54
48
10
32
92
10

Expected Output:
32 48 92 54 10 10
54 48 32 92 10 10
32 92 48 10 10 54
```

3. (30 points) Write a program to convert an arithmetic expression given in postfix notation into an expression tree. Print the inorder traversal of the resulting tree.

Input-Output Format: The input consists of a single line containing the input postfix expression. Assume that the input is a valid postfix expression where each operand is represented by a single lower case letter and each of the operators come from the set $\{+, -, *, /\}$. The output consists of a single line containing the inorder traversal sequence of the corresponding expression tree.

Question 3 continues on the next page...

```
Sample Input:
xy*z-pq+*

Expected Output:
x*y-z*p+q
```