Indian Institute of technology, Guwahati
Department of Computer Science and Engineering
Data Structure Lab (CS210) Assignment: 1

Date: 13th August 2018. **Total Marks: 40**

1. Creates Binary tree of positive distinct integers from the input sequence given in a specific format: <**number1 parent1 L/R**>. You need to insert the key number1 in L/R position of the node with key parent1. All numbers are unique and non-negative. If a number (let's say x) is asked to be inserted at a position where some number (let's say y) is already present, then print a warning message "**y is already present at the specified position for x**" and ignore inserting x in the tree. Initially tree is empty, you need to make root node containing number1 of first tuple <number1 parent1 L/R> you encounter ignoring L/R and parent1. **(10)**
2. Prints in-order traversal of the created tree using stack and without any recursive function. Also, you need to print the maximum depth of the stack during the traversal. **(10)**
3. Write a function to find the height of the tree (height of the root is 0). **(Bonus Question: 5)**
4. Creates a right threaded Binary tree from the same input. Again, if a number (let's say x) is asked to be inserted at a position where some number (let's say y) is already present then ignore inserting x in the tree and carry on inserting other numbers. **(10)**
5. Print in-order traversal of the created threaded tree using threads. During traversal, you encounter a node with right thread, then print T after printing that number of the node**.** **(10)**

Input format:

```
Number of lines
number parent L/R
…..
```

Example Input for the tree given in fig. 1:

```
12
3   -1   L
6    3   R
4    6   L
2    3   L
19  -1   L
8    6   R
10   6   L
7    8   L
5    4   R
59   8   L
9    8   R
1    2   L
```

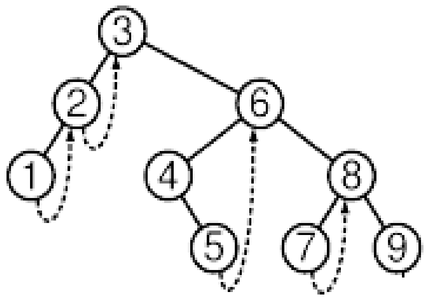

Figure 1: Sample Tree

Output Format:

```
Warnings (if any)
Inorder_traversal
Max stack depth
Height_of_tree          #bonus Question
Inorder traversal order for threaded binary tree
```

Example Output for the above tree:

```
3 is already present at specified position for 19
4 is already present at specified position for 10
7 is already present at specified position for 59
1 2 3 4 5 6 7 8 9
3
3
1T  2T  3  4  5T  6  7T  8  9
```

Evaluation Guidelines:

1. Full marks if both methods work for all test cases.
2. If code is not working but most of code is written, maximum 40% can be given based on TA's evaluation.
3. 20% marks will be deducted for each test not running.
4. At most 10% marks will be deducted for bad coding style. i.e., (1) code is not modular (2) code is not properly indented (3) code is not properly commented and (4) Variable and functions are not suitably named.
5. TA will help you initially to get rid of segmentation fault and compilation error, etc. in your code. TA will not help you find out the solution of the assignment given.

Test Cases

Test case 1:

Input:

```
7
6  -1   L
8   6   L
2   8   L
4   8   L
1   2   L
3   8   L
5   1   L
```

<u>Output:</u>

```
2 is already present at specified position for 4
2 is already present at specified position for 3
5 1 2 8 6
5
4
5T 1T 2T 8T 6
```

<u>Test case 2:</u>
<u>Input:</u>

```
7
9    -1     L
4     9     R
6    -1     R
8     4     R
1     8     R
5     4     R
2     1     R
```

<u>Output:</u>

```
9 is already present at specified position for 6
8 is already present at specified position for 5
9 4 8 1 2
1
4
9 4 8 1 2
```

<u>Test case 3:</u>
<u>Input:</u>

```
10
27    -1    L
35    27   R
42    35   R
26    27   R
14    27   L
51    42   R
31    35   L
10    14   L
6      42   R
```

```
43    51   L
```

Output:

```
35 is already present at specified position for 26
51 is already present at specified position for 6
10 14 27 31 35 42 43 51
3
4
10T 14T 27 31T 35 42 43T 51
```

Test case 4:

Input:

```
11
99    -1    L
11    99   R
91    99   L
12    99   R
88    91   R
4      -1    L
15    11   L
77    88   L
4      77   L
67    15   R
60    67   L
```

Output:

```
11 is already present at specified position for 12
99 is already present at specified position for 4
91 4 77 88 99 15 60 67 11
4
4
91 4T 77T 88T 99 15 60T 67T 11
```