Coding Assignment 2: CS2233

September 16, 2024

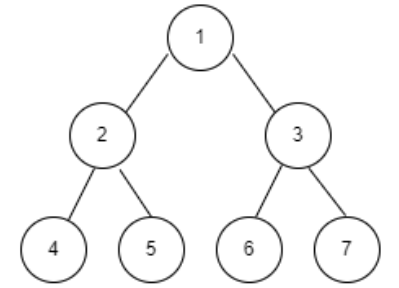
Kindly adhere to the following instructions.

- Please write a C/C++ program corresponding to each problem. Your code should be well commented on, and variable names should be appropriately chosen. Also, prepare a `readme` text file where you can mention instructions to run the program/how to take input, etc.
- Create a folder and put all the code files and `readme` text file in it, give a name to the folder as "yourName_yourRollNo", zip the folder and submit it to the Google Classroom portal.
- Strictly follow the input and output format for each problem.
- Any code that does not follow the input-output criteria won't be evaluated and will get **ZERO**.
- Your code will also be checked against plagiarism (both from web and peer).
- Any form of plagiarism (web/chatGPT/with peers) will be severely penalised and will result in an F grade.
- The submission (strict) timeline is 30th September, Monday, 11 AM.
- Each question consists of 10 marks.

General Instruction

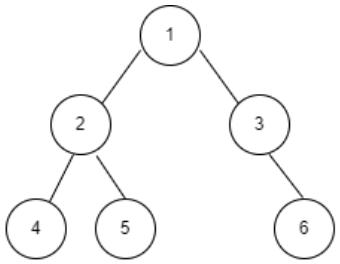
The description of the *array representation of a tree* is as follows:

- Array index starts with 1.
- The left and right children of the i^{th} node present in the $2i^{th}$ and $(2i+1)^{th}$ index, respectively.
- If let's say left child of the i^{th} node is empty then the $2i^{th}$ index of array contains $NULL$, similarly for the right child also.



1	2	3	4	5	6	7

Array representation of the above tree



1	2	3	4	5	NULL	6

Array representation of the above tree

- For example, See the above images.

---

1. **Write a non-recursive implementation of `inorder, preorder, postorder` traversal.**

   Input format

   - First line will contain k, which indicates the number of test cases.
   - Following k lines will contain arrays of integers, denoting the array representation of a tree.

   Output format

   - Your output also should contain $3k$ lines, which indicates the output of corresponding k inputs.
   - For each input, first line contains the `inorder` traversal, second line contains the `preorder` traversal, and third line contains the `postorder` traversal.

   Example:
   Input:

   ```
   2
   1 2 3 4 5 6 7
   1 2 3 4 5 NULL 6
   ```

   Output:

```
4 2 5 1 6 3 7   %inorder traversal of 1st tree%
1 2 4 5 3 6 7   %preorder traversal of 1st tree%
4 5 2 6 7 3 1   %postorder traversal of 1st tree%
4 2 5 1 3 6     %inorder traversal of 2nd tree%
1 2 4 5 3 6     %preorder traversal of 2nd tree%
4 5 2 6 3 1     %postorder traversal of 2nd tree%
```

Note: %comments% are there for your understanding, you need not print this.

2. **Write a C program that takes `inorder` and `preorder` traversal as input, output the tree. You need to print the array representation of the tree. Your code should output an error message if the `inorder` and `preorder` are not corresponding to the same tree.**

   Input format

   - First line will contain k, which indicates the number of test cases.
   - Following $2k$ lines will contain integers of each $2k$ arrays.
   - $(2i-1)^{th}$ line contains the `inorder` traversal for the i^{th} test case.
   - $2i^{th}$ line contains the `preorder` traversal for the i^{th} test case.

   Output format

   - Your output also should contain k lines, which indicates the output of corresponding k tree inputs.
   - Each line will contain the array representation of the tree as described in the general instruction.
   - If the `inorder` and `preorder` are not corresponding to the same tree, then print `ERROR`.

   Example:
   Input:

```
3
4 2 5 1 6 3 7   %inorder traversal of 1st tree%
1 2 4 5 3 6 7   %preorder traversal of 1st tree%
4 2 5 1 3 6     %inorder traversal of 2nd tree%
1 2 4 5 3 6     %preorder traversal of 2nd tree%
4 2 6 1 3 5     %inorder traversal of 3rd tree%
1 2 4 5 3 6     %preorder traversal of 3rd tree%
```

   Output:

```
1 2 3 4 5 6 7
1 2 3 4 5 NULL 6
ERROR
```

Note: %comments% are there for your understanding.

3. **Write a C program that takes `inorder` and `postorder` traversal as input, output the tree. You need to print the array representation of the tree. Your code should output an error message if the `inorder` and `postorder` are not corresponding to the same tree.**

Input format

- First line will contain k, which indicates the number of test cases.
- Following $2k$ lines will contain integers of each $2k$ arrays.
- $(2i-1)^{th}$ line contains the `inorder` traversal for the i^{th} test case.
- $2i^{th}$ line contains the `postorder` traversal for the i^{th} test case.

Output format

- Your output also should contain k lines, which indicates the output of corresponding k tree inputs.
- Each line will contain the array representation of the tree as described in the general instruction.
- If the `inorder` and `postorder` are not corresponding to the same tree, then print `ERROR`.

Example:
Input:

```
3
4 2 5 1 6 3 7   %inorder traversal of 1st tree%
4 5 2 6 7 3 1   %postorder traversal of 1st tree%
4 2 5 1 3 6     %inorder traversal of 2nd tree%
4 5 2 6 3 1     %postorder traversal of 2nd tree%
4 2 6 1 3 5     %inorder traversal of 3rd tree%
4 5 2 6 3 1     %postorder traversal of 3rd tree%
```

Output:

```
1 2 3 4 5 6 7
1 2 3 4 5 NULL 6
ERROR
```

Note: %comments% are there for your understanding.

4. **Write a C program that takes `preorder` and `postorder` traversal as input, output the tree. It is given that each node consists of exactly two children. Your code should output an error message if the `preorder` and `postorder` are not corresponding to the same tree. You need to print the the array representation of the tree.**

   Input format

   - First line will contain k, which indicates the number of test cases.
   - Following $2k$ lines will contain integers of each $2k$ arrays.
   - $(2i-1)^{th}$ line contains the `preorder` traversal for the i^{th} test case.
   - $2i^{th}$ line contains the `postorder` traversal for the i^{th} test case.

   Output format

   - Your output also should contain k lines, which indicates the output of corresponding k tree inputs.
   - Each line will contain the array representation of the tree as described in the general instruction.
   - If the `preorder` and `postorder` are not corresponding to the same tree, then print `ERROR`.

   Example:
   Input:

```
3
1 2 4 5 3 6 7     %preorder traversal of 1st tree%
4 5 2 6 7 3 1     %postorder traversal of 1st tree%
3 7 8 9 10        %preorder traversal of 2nd tree%
7 9 10 8 3        %postorder traversal of 2nd tree%
1 2 4 5 7 6 3     %preorder traversal of 3rd tree%
4 7 6 5 3 2 1     %postorder traversal of 3rd tree%
```

   Output:

```
1 2 3 4 5 6 7
3 7 8 NULL NULL 9 10
ERROR
```

   Note: %comments% are there for your understanding.

5. **Write a C program that takes an arithmetic tree as input and outputs the result of the arithmetic expression. The leaf node is numeric data in an arithmetic expression tree, and the `non-leaf/internal` node is the operator.**

   Input format

   - First line will contain k, which indicates the number of test cases.
   - Following k lines will contain node value of tree, denoting the array representation of a tree.

   Output format

   - Your output also should contain k lines, which indicates the output of corresponding k inputs.
   - For each input, Print the result of arithmetic tree as output.

   Example:
   Input:

```
3
+ * / 5 4 4 2
* 2 + NULL NULL 2 4
/ - 5 + * NULL NULL 10 2 1 2 NULL NULL NULL NULL
```

   Output:

```
22
12
2
```

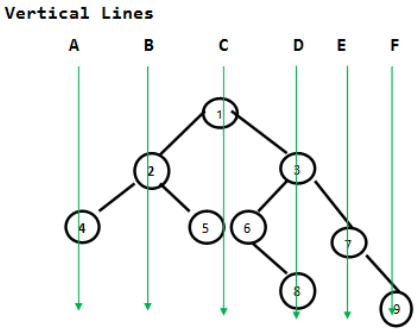
6. **Write a C program that prints the given tree vertically. The following example explains the vertical tree traversal.**

   Input format

   - First line will contain k, which indicates the number of test cases.
   - Following k lines will contain arrays of integers, denoting the array representation of a tree.

   Output format

   - For i^{th} input, first print "**Tree :** i ".
   - From the next line, print the corresponding output.
   - Your output should contain 1 line for each vertical level.



Vertical order traversal is:

A- 4

B- 2

C- 1 5 6

D- 3 8

E- 7

F- 9

Example
Input:

```
4
1 2 3
1 2 3 NULL 4 NULL 5
1 2 3 4 NULL 5 6
1 2 3 4 5 NULL NULL 6 NULL NULL 7 NULL NULL NULL NULL
```

Output:

```
Tree : 1
2
1
3    %vertical traversal of the tree%
Tree : 2
2
1 4
3
5    %vertical traversal of the tree%
Tree : 3
4
2
1 5
3
6    %vertical traversal of the tree%
```

```
Tree : 4
6
4
2
1 5
3 7    %vertical traversal of the tree%
```

Note: %comments% are there for your understanding, you need not print this.