CS241 – Data Organization
Fall 2022

Programming Assignment #8:
Recursive Binary Tree Functions
Due by 11:59 PM on Wednesday, November 30, 2022

In this class we have seen how the basic depth-first-search, tree-hugging traversal of a Binary Tree can be written very compactly as a recursive function. By augmenting this basic traversal algorithm, we can perform a variety of different computations on a binary tree. For this assignment you must implement the following binary tree functions.

```
bool  isHeapOrdered ( Node * curr );
int   numSingleChild ( Node * curr );
void  makeMirror ( Node * curr );
```

1. Assignment

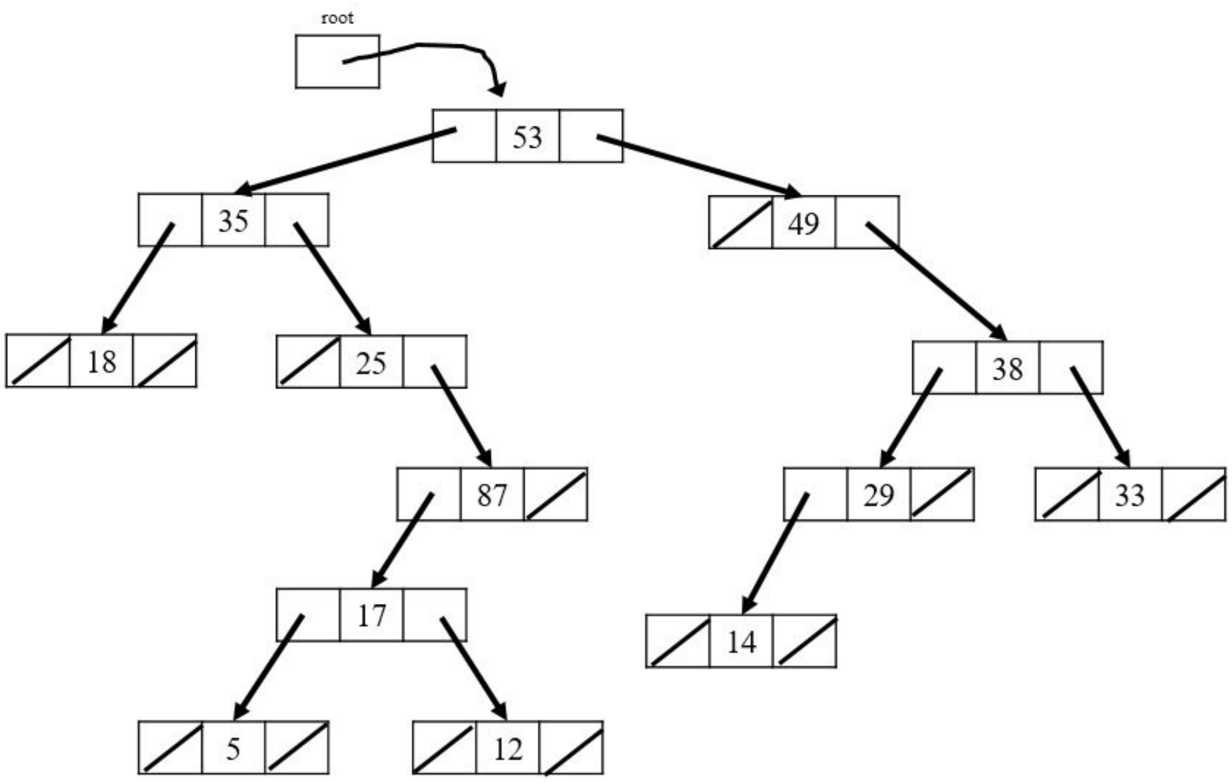
The provided source code contains a **stub** for each of these functions, together with a description of what the function should do. You must complete the **body** of each of these three functions.

The **`isHeapOrdered`** function returns true or false depending on whether the binary tree is max-heap-ordered (i.e., if every node's value is greater-than-or-equal-to the values of its children). Note that we are **not** concerned at all about the shape of the tree. A tree can be heap-ordered even if it does not have a "complete" shape.

The **`numSingleChild`** function returns the number of nodes in the tree that have exactly one child. For example, in the tree below, the function would return the value 4.

The **`makeMirror`** function alters the shape of the tree (ie, it alters the left and right child pointers) so that the tree is now a mirror-image of the former shape. In other words, the tree has been "flipped" through an imaginary vertical line through the root of the tree.

You should implement each of these functions in the **<u>most efficient manner</u>** possible. The given source code in file `prog08_HANDOUT.c` includes functions `size`, `contains`, `preorder`, and `postorder` that we have studied in this class. It also includes the declarations of the global variables you need, and a set of functions that we use for building the tree.



The main function creates and initializes a pointer to a BinaryTree struct called `myTree` which handles the information of the tree. The `prog08_HANDOUT.c` file provided processes from standard input the information related to tree contained in an input file (we provide three input files to test) and builds the tree.

Your program should read a list of operations/queries from the command line and using the `root` pointer of the `myTree` variable mentioned above, process each one of those operations by calling the appropriate functions. Each command may ask you to print some piece of information to standard output which should always be followed a by new line character ('\n'). You will invoke the program as follows:

```
$  ./a.out   list-of-desired-operations  <  inputFileName
```

The syntax for the possible operations/queries in the list-of-desired-transformations is as follows:

- **isordered** –Your program should output "YES" if root node of `myTree` is heap ordered as defined above, otherwise your program should print "NO".
- **numsinglechild** – Your program should print the result of calling the function `numSingleChild` using the root node of `myTree` as argument.
- **makemirror** – Your program should call the function `makeMirror` using the the root node of `myTree` as argument. No output is requested after completing this operation.

- **preorderprint** - Your program should call the function `preorder` (provided) using the root node of `myTree`.
- **postorderprint -** Your program should call the function `postorder` (provided) using the root node of `myTree.`
- **contains –** This operation is followed by one integer n. Your program should output "YES" if the root of `myTree` contains a node with data n, otherwise your program should print "NO" (the code for `contains` is provided).

  For example, the program might be invoked as:

```
$ ./a.out makemirror preorderprint makemirror postorderprint <   inputFileName
```

If the arguments in the list-of-desired-operations are not valid commands as defined above, your program should exit immediately with `exit(2);`

We provide `inputFile1, inputFile2` and `inputFile3` for testing purposes.

2. Your tasks

In summary, your tasks for this project are:

1. Build appropriate and efficient code for the three functions: `isHeapOrdered, numSingleChild, makeMirror.`
2. Build code in main for parsing the command line and calling the appropriate functions that will execute the operations specified in the command line.

That is it! To be successful in this project, you first need to read this whole statement (yes, read it all before you begin), ask questions if you need to; then download the files provided (three input files and the `prog08_HANDOUT.c`) on canvas or copy them from the directory indicated on canvas, study the .c provided to become familiar with the declarations, its structure and where to include the code that you will be writing to complete the two tasks listed above.

3. What to submit

You should be sure to include **THE NAME OF THE AUTHOR OF THE PROGRAM** in a comment at the top of your source code file, for this and all other assignments in this course.

Your source code must use proper **style**, that is variables should be well named (name is not too short, not too long, and is meaningful), and bodies of loops, if's, etc.. should be properly indented. Refer to the coding style file for this class, published on Learn under Coding Standards in a file named: cs241_codingStandards2020.pdf.

Create a .c file for this assignment and name it using your last name and the initial of your first name, like this: **lastName_initialFirstName_RBTF.c**

Submit this file for grading to canvas in the place of this assignment.

4. General rubric

If any of your C programs do not compile with the -ansi -pedantic -Wall options without errors, the points given for the assignment will be zero. Otherwise, the following rubric will be used:

- +10 pts: Your C file follows the class coding style.
- +10 pts: Your program correct process the command line arguments.
- +55 pts: Your program provides correct implementation of the functions `isHeapOrdered`, `numSingleChild`, `makeMirror`.
- +15 pts: Your program matches the output of the following executions of your program:
    - ./a.out preorderprint makemirror postorderprint < inputFile1
    - ./a.out postorderprint makemirror numsinglechild < inputFile2
    - ./a.out contains 2 isordered contains 1 < inputFile3
- +10 pts: Your program matches the output of internal files only available to the grader.