

**CISS350: Data Structures and Advanced Algorithms**  
**Quiz q10702**

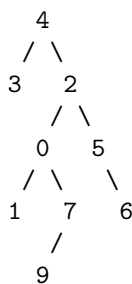
Name: YOUR EMAILScore: 

You are given the following binary tree node class:

```
class BTreeNode
{
public:
    int key_;
    BTreeNode * parent_, * left_, * right_;
};
```

The following questions uses this class. Make sure you test your code.

Q1. In the following tree

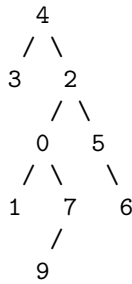


assume that the tree contains unique values. Suppose **p** points to the node with value 4. Then the tree that **p** points to (i.e., the root of the tree that **p** points to) has height 4. If **p** points to the node with value 2. Then the tree that **p** points to has height 3. Complete the following function that returns the height as described above.

ANSWER:

```
int height(BTreeNode * p)
{
}
```

Q2. In the following tree



assume that the tree contains unique values.

- Suppose **p** points to the node with value 4 and **q** points to the node with value 5. Then we say that 5 is at a depth of 2 below 4 and when I call `depth(p, q)`, I get 2.
- If **p** points to the node with value 2 and **q** points to the node with value 9. Then we say that 9 is at a depth of 3 below 2 and when I call `depth(p, q)`, I get 3.
- If **p** points to the node with value 2 then `depth(p, p)` returns 0.

The above describes the cases when **p** and **q** points to nodes and either **p** and **q** are the same or **\*q** is a descendent of **\*p**. In all other cases, the function should return `-1`. Complete the following function that returns the depth as described above.

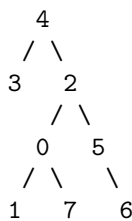
ANSWER:

```

int depth(BTNode * p, BTNode * q)
{
}

```

Q3. In the following tree



assume that the tree contains unique values.

- Suppose **p** points to the node with value 4, the depth of 0 (below 4) is said to be 2. The depth of 6 (below 4) is 3.

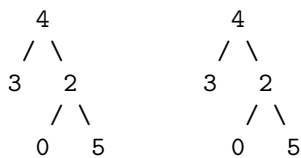
Check the notes. Complete the following function that returns the depth of the node with `key_` value of `target` below the node that **p** points to. If `target` cannot be found, `-1` is returned.

ANSWER:

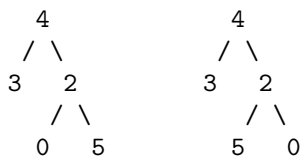
```
int depth(BTNode * p, int target)
{
}
```

(Hint: Do a BF traversal. Instead of putting node pointers into the queue, you want to put the node pointer together with its depth into the queue. Note that even if the tree has repeat **key\_** values, if you do a BF traversal, the depth computed is the *shallowest* depth of all the nodes with the **key\_** value of **target**.)

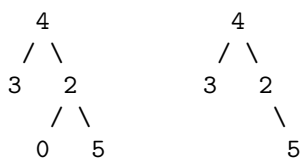
Q4. Complete the function below that returns true if **p** and **q** point to roots of binary trees which are the same. In this case, same means the tree has the same structure, i.e., the root nodes have the same **key\_** value, while left child nodes have the same value, then right child node have the same **key\_** value, etc. And if a node in the tree **p** points to does not have a left child, then the corresponding node in the tree **q** points to does not have a left child; likewise for right child. The following trees are the same



The following trees are different:



The following trees are different:



Don't forget to check the parent pointer.

ANSWER:

```
bool is_same_tree(BTNode * p, BTNode * q)
{
}
```

## INSTRUCTIONS

In the file `thispreamble.tex` look for

```
\renewcommand\AUTHOR{}
```

and enter your email address:

```
\renewcommand\AUTHOR{jdoe5@cougars.ccis.edu}
```

(This is not really necessary since alex will change that for you when you execute `make`.) In your bash shell, execute “`make`” to recompile `main.pdf`. Execute “`make v`” to view `main.pdf`.

Enter your answers in `main.tex`. In the bash shell, execute “`make`” to recompile `main.pdf`. Execute “`make v`” to view `main.pdf`.

For each question, you’ll see boxes for you to fill. For small boxes, if you see

```
1 + 1 = \answerbox{}
```

you do this:

```
1 + 1 = \answerbox{2}
```

`answerbox` will also appear in “true/false” and “multiple-choice” questions.

For longer answers that need typewriter font, if you see

```
Write a C++ statement that declares an integer variable name x.  
\begin{answercode}  
\end{answercode}
```

you do this:

```
Write a C++ statement that declares an integer variable name x.  
\begin{answercode}  
int x;  
\end{answercode}
```

`answercode` will appear in questions asking for code, algorithm, and program output. In this case, indentation and spacing is significant. For program output, I do look at spaces and newlines.

For long answers (not in typewriter font) if you see

```
What is the color of the sky?  
\begin{answerlong}  
\end{answerlong}
```

you can write

```
What is the color of the sky?  
\begin{answerlong}  
The color of the sky is blue.  
\end{answerlong}
```

A question that begins with “T or F or M” requires you to identify whether it is true or false, or meaningless. “Meaningless” means something’s wrong with the question and it is not well-defined. Something like “ $1 + 2 = 4$ ” is either true or false (of course it’s false). Something like “ $1+2 = 4?$ ” does not make sense.

When writing results of computations, make sure it’s simplified. For instance write 2 instead of  $1 + 1$ .

#### HIGHER LEVEL CLASSES.

For students beyond 245: You can put L<sup>A</sup>T<sub>E</sub>X commands in `answerlong`.

More examples of meaningless statements: Questions such as “Is  $42 = 1+2$  true or false?” or “Is  $42 = \{2\}^{\{3\}}$  true or false?” does not make sense. “Is  $P(42) = \{42\}$  true or false?” is meaningless because  $P(X)$  is only defined if  $X$  is a set. For “Is  $1 + 2 + 3$  true or false?”, “ $1 + 2 + 3$ ” is well-defined but as a “numerical expression”, not as a “proposition”, i.e., it cannot be true or false. Therefore “Is  $1 + 2 + 3$  true or false?” is also not a well-defined question.

More examples of simplification: When you write down sets, if the answer is  $\{1\}$ , do not write  $\{1, 1\}$ . And when the values can be ordered, write the elements of the set in ascending order. When writing polynomials, begin with the highest degree term.

When writing a counterexample, always write the simplest.