CSc 120: Intro to Computer Programming II

Spring 22 (Lewis)

Test 2

Fri 1 Apr 2022

Solutions

Name: ______________________________ NetID: _______________

Question	Points	Score
Short Answer	20	
True or False?	10	
Short Algos	22	
BST Errors	9	
A Linked List Algorithm	15	
BST Actions	14	
Reference Diagram	10	
Total:	100	

1. (a) (5 points) Explain why every method in a class needs a `self` parameter.

   Solution: You need to know what object you are accessing or modifying!

   (b) (5 points) Why was recursion so important for writing algorithms on trees? What's wrong with loops?

   Solution: In a tree, the structure branches out in multiple directions from each node. Thus our simple while loop, which used a single `cur` pointer, doesn't work, as we cannot go back in time,when we hit a leaf, and choose to follow an alternate path.

   (c) (5 points) Why is it not possible to traverse a singly-linked list backwards?

   Solution: References only go one way! So we can follow the 'next' pointer, but we cannot go backwards, to find the node that points at cur.

   (d) (5 points) The following attempts to print out every third value in a linked list, but it occassionally crashes.. Explain what the bug is, and how to fix it.

```
head = ...
cur = head
while cur is not None:
    print(cur.val)
    if cur.next is None:
        break
    cur = cur.next.next.next
```

   Solution: We use `cur.next.next` withoutcheceking to see if it is None. Add a second guard condition to the `if` statement to fix this.

2. (10 points) **True or False?**

Arrays and linked lists are the same thing.
○ True [✓ **False**]

To check if a tree follows the BST ordering rule, you simply need to compare each node to its children: if the left child is less, and the right child is more, then it is a BST.
○ True [✓ **False**]

> **Solution: Explanation:**
> The given rule only compares a parent to its immediate children. In order to fulfill the BST ordering rule, we have to compare the parent to every descendant in each subtree.

Python's built-in `for` loop doesn't work on Linked Lists.
[✓ **True**] ○ False

Deleting a value from the middle of an array takes $O(1)$ time.
○ True [✓ **False**]

All BSTs are balanced.
○ True [✓ **False**]

NetID: ____________________

3. (a) (11 points) Write a function that will print the in-order traversal of a binary tree, but **reversed.** Print one value per line.

   You **must not** use a helper function.

```
def in_ord_rev(root):
```

Solution:

```
    if root is None:
        return

    in_ord_rev(root.right)
    print(root.val)
    in_ord_rev(root.left)
```

   (b) (11 points) Write a recursive function which takes an array as input, and prints the values of the inputs in pairs. The **last** line of output should contain both the **first and last** values from the array; the second-to-last line should contain the second and second-to-last, etc. (If the number of elements in the array is odd, then **do not print** the one in the middle.)

   Example:

```
INPUT:
  [ 1,2,3,4,5 ]
OUTPUT:
  2 4
  1 5
```

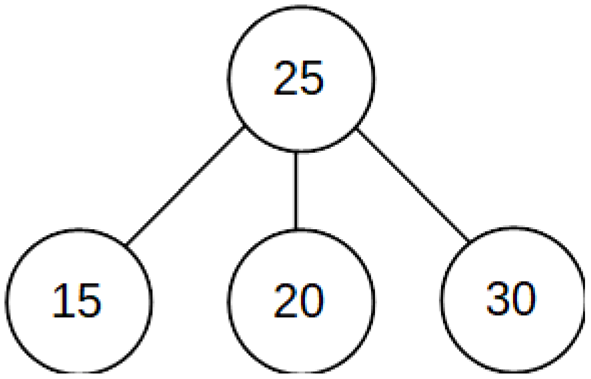
```
def fold(data):
```

Solution:

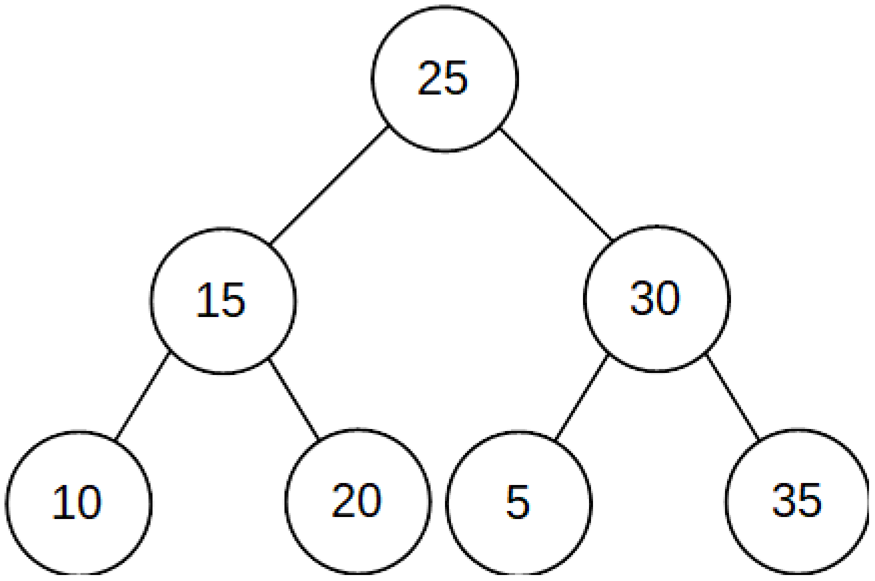
```
    if len(data) < 2:
        return

    fold([1:-1])
    print(fold[0], fold[-1])
```

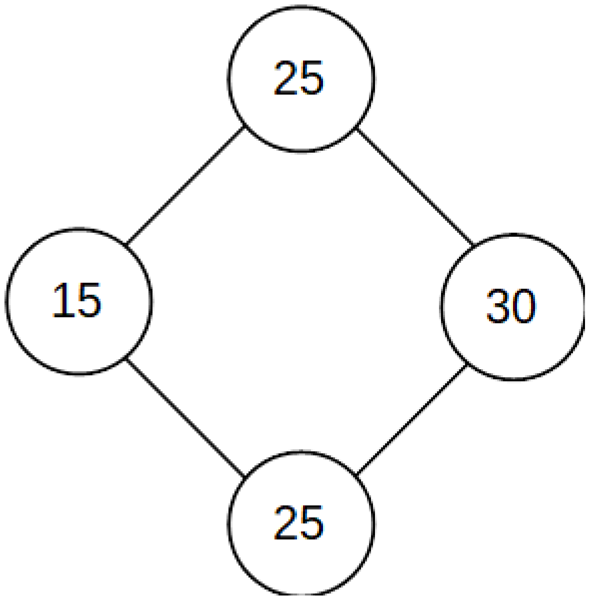
4. (9 points) Each of these figures is **not** a BST. For each one, give one reason why it is not a BST (there might be more than one reason, only give us one for each.)



Solution: The root node has three children.



Solution: The 5 is out of place; it must **not** be on the right side of 25.



Solution: This is a diamond shape; we cannot have more than one path from the root to any given node.

NetID: ____________________

5. (15 points) Write a function that takes a linked list as the only parameter; you may assume that the list has at least two nodes, and that all of the values are numeric. Return the maximum difference between any two consecutive nodes. For example, if your input list was

```
17 -> 5 -> 0 -> 6 -> -4 -> 7
```

you should return 12, since the largest difference was between 17 and 5.

NOTE: Python's built-in `abs()` function will help you.

Solution:

```
def max_diff(head):
    # we were told that the values were numeric, and we are only
    # checking absolute value (no negatives), so 0 is an excellent
    # starting value.
    retval = 0

    while cur.next is not None:
        this_diff = abs(cur.val - cur.next.val)
        if this_diff > retval:
            retval = this_diff

    return retval
```

Instructor's Note:
I didn't expect recursive solutions to this problem (other than a few), but I'm seeing many of them. Good job! Here's my recursive solution:

```
def max_diff(head):
    this_diff = abs(head.val - head.next.val)

    if head.next.next is None:
        return this_diff
    else:
        return max(this_diff, max_diff(head.next))
```

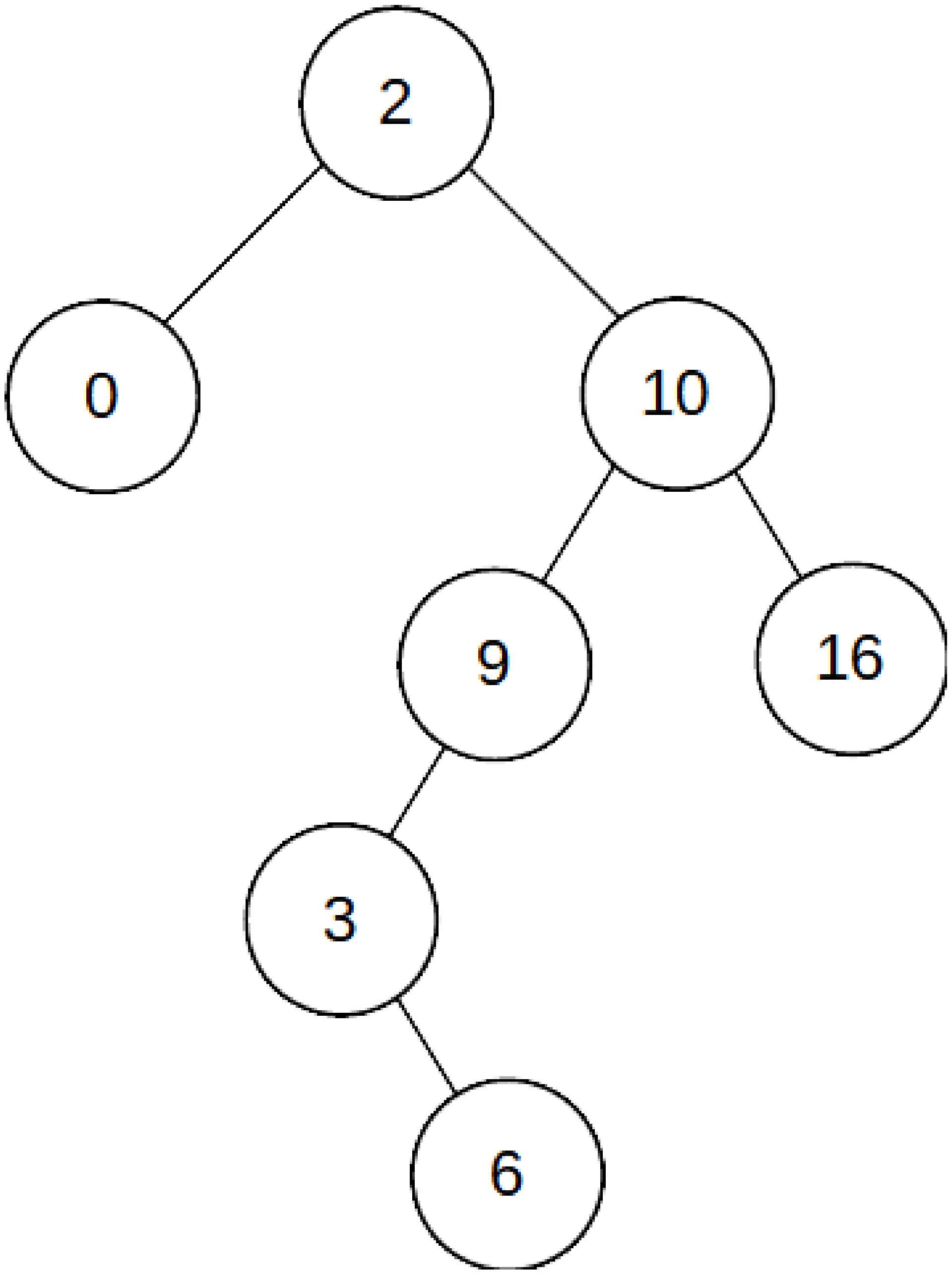
An Interesting Student Solution:
The function definition says that you will never have to deal with lists shorter than two; I did that because, in a shorter list, the idea of "difference between two nodes" is kind of meaningless. But this student came up with a meaningful return value - which incidently would **always** be less than any real difference. So it's a wonderful base case.

```
def max_diff(head):
    if head is None or head.next is None:
        return -1
    return max( abs(head.val-head.next.val), max_diff(head.next) )
```

6. (a) (7 points) Insert the following values into a BST. Start with an empty tree, and then insert the values in the order specified. Draw the BST only once; don't draw all of the steps you had to take to build the tree over time.

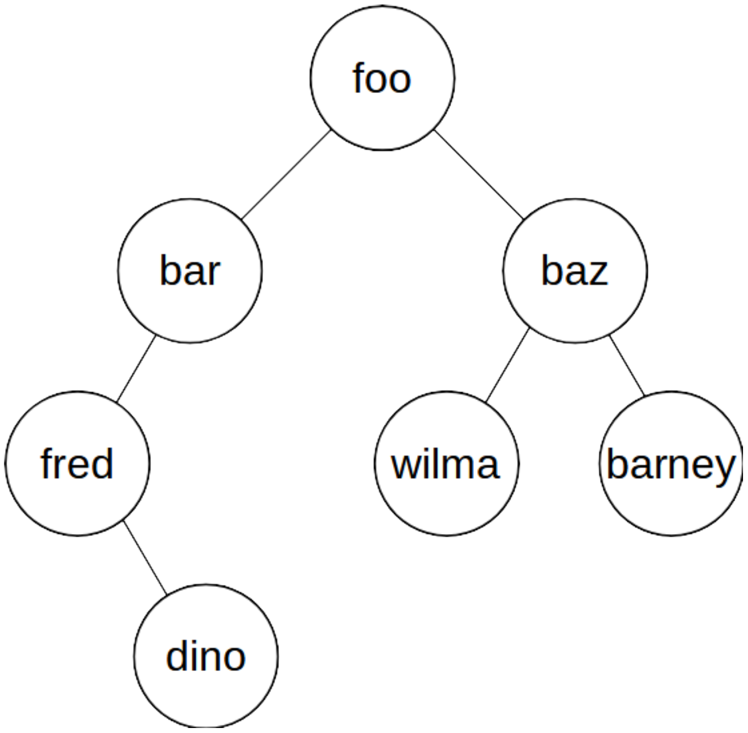
   2 0 10 9 3 6 16

Solution:



(b) (7 points) Give an in-order traversal, and then a pre-order traversal of the following tree. **Make sure to label which is which, carefully.**

NetID: ____________

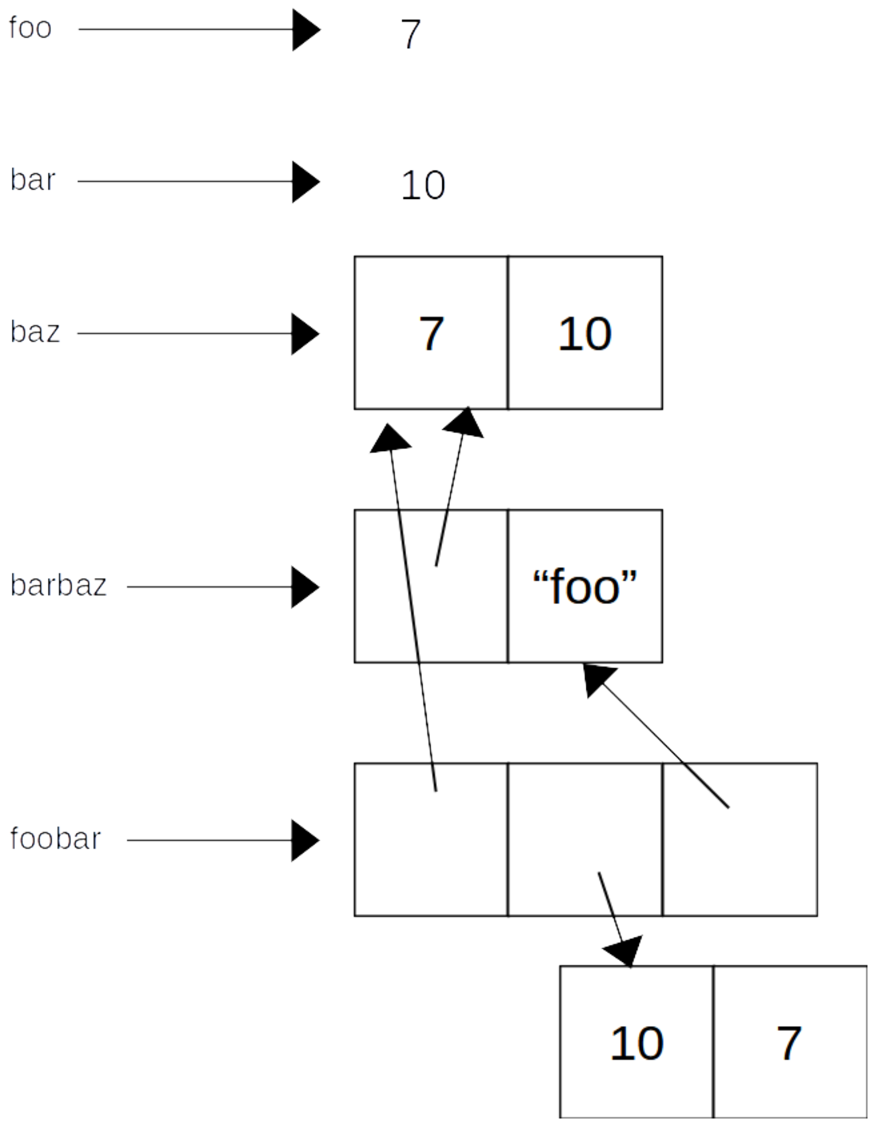


Solution: In-Order:
fred, dino, bar, foo, wilma, baz, barney

Pre-Order:
foo, bar, fred, dino, bar, wilma, barney

7. (10 points) Draw a reference diagram of the objects produced by the following code. Only draw a single diagram, but your diagram should include all five of the variables (`foo, bar, baz, barbaz, foobar`).

```
foo = 7
bar = foo+3
baz = [foo,bar]
barbaz = [baz, "foo"]
foobar = [baz, [bar,foo], barbaz]
```



Solution: