3013 - Algorithms
Spring 2023
Exam 2
March 24 2023

Name: ____________________

READ THESE INSTRUCTIONS
• Use pencil only • Put name on all pages if separated. • Do not remove the staple from your exam. So, don't separate them:) • Do not crumple or fold your exam. • Handwriting that is illegible (messy, small, not straight) will lose points. • Indentation matters. Keep code aligned correctly. • **Answer all questions on the answer sheet(s) provided**. • It may be the test itself, or it may be sheets given to you separately. If you are not sure ask. But I can assure you it is either one or the other and not a mixture of both. • Help me ... help you!
Failure to comply will result in loss of letter grade

1. MULTIPLE CHOICE

1. (3 points) What is an AVL tree?
   - **A. A binary tree with a balance factor of -1, 0, or 1.**
   - B. A binary tree with a balance factor of -2, 0, or 2.
   - C. A binary tree with a balance factor of -3, 0, or 3.
   - D. A binary tree with no balance factor.
2. (3 points) What is the maximum height difference allowed between the left and right subtrees in an AVL tree?
   - **A. 1**
   - B. 2
   - C. 3
   - D. It can be any height difference.
3. (3 points) What is the time complexity of AVL tree operations such as insertion, deletion, and searching in the worst case scenario?
   - A. O(1)
   - B. O(n)
   - **C. O(log n)**
   - D. O(n log n)
4. (3 points) Which of the following operations can unbalance an AVL tree?
   - A. Insertion
   - B. Deletion
   - **C. Both insertion and deletion**
   - D. Neither insertion nor deletion
5. (3 points) What is the AVL tree's primary advantage over a regular binary search tree?
   - **A. AVL tree's are always balanced, while binary search tree's may not be.**
   - B. AVL tree's can handle any type of data, while binary search tree's only handle numerical data.
   - C. AVL tree's can handle any number of nodes, while binary search tree's have a limit.
   - D. AVL tree's have no advantage over binary search tree's.
6. (3 points) Which traversal visits the nodes in the following order: left subtree, root, right subtree?
   - **A. Inorder traversal**
   - B. Preorder traversal
   - C. Postorder traversal
   - D. None of the above
7. (3 points) Which traversal visits the nodes in the following order: left subtree, right subtree, root?
   - A. Inorder traversal

B. Preorder traversal

C. Postorder traversal

D. None of the above

8. (3 points) Which traversal visits the nodes in the following order: root, left subtree, right subtree?

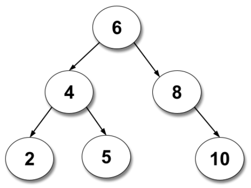
A. Inorder traversal

B. Preorder traversal

C. Postorder traversal

D. None of the above

9. (3 points) What is the order of nodes visited in a postorder traversal of the following binary search tree?



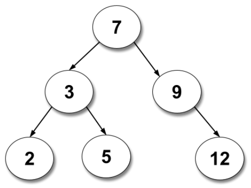
A. 2, 5, 4, 10, 8, 6

B. 2, 5, 4, 6, 8, 10

C. 5, 2, 4, 10, 8, 6

D. 5, 2, 10, 4, 8, 6

10. (3 points) What is the order of nodes visited in a preorder traversal of the following binary search tree?



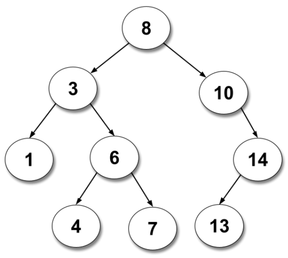
A. 7, 3, 2, 5, 9, 12

B. 2, 3, 5, 7, 9, 12

C. 7, 9, 12, 3, 5, 2

D. 2, 5, 3, 7, 9, 12

Use this Binary Search Tree to answer the next two question:



11. (3 points) What is the inorder successor of the node with value 6?
    - A. 1
    - B. 3
    - C. 4
    - **D. 7**
    - E. 8
    - F. 10

12. (3 points) What is the inorder predecessor of the node with value 10? Think!
    - A. 1
    - B. 3
    - C. 4
    - D. 7
    - **E. 8**
    - F. 10

13. (3 points) What is the minimum number of child nodes an inner node can have in a binary search tree?
    - A. 0
    - **B. 1**
    - C. 2
    - D. None of the above

14. (3 points) What is the inorder successor of a node in a binary search tree?
    - **A. The node with the next largest value in the tree**
    - B. The node with the next smallest value in the tree
    - C. The root of the tree
    - D. The node with the same value as the current node

15. (3 points) What is the load factor of a hash table?
    - A. The number of elements stored in each bucket
    - **B. The ratio of the number of elements in the table to the number of buckets in the table**
    - C. The average number of collisions per operation
    - D. The time it takes to perform operations on the table

16. (3 points) What is a good choice for the load factor of a hash table?
    - A. 0.1
    - B. 0.5
    - **C. 0.7-0.8**
    - D. 1.0

17. (3 points) How does the load factor affect the performance of a hash table?
    - A. A higher load factor can increase the likelihood of collisions and reduce efficiency
    - B. A lower load factor can result in wasted space and increased memory usage

C. Both A and B

D. Neither A nor B

18. (3 points) How is the size of a hash table chosen?

A. Based on the number of elements to be stored and the desired load factor

B. Based on the type of data being stored

C. Based on the number of buckets in the hash function

D. Based on the size of the input data

19. (3 points) What happens if the size of a hash table is too small?

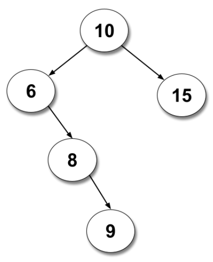
A. It can result in a higher likelihood of collisions and reduced efficiency

B. It can result in wasted space and increased memory usage

C. Both A and B

D. Neither A nor B

20. (3 points) Which node should be rotated and in which direction to balance the tree using a single right rotation?



A. Node 8 should be rotated to the right.

B. Node 8 should be rotated to the left.

C. Node 10 should be rotated to the right.

D. Node 10 should be rotated to the left.

21. (3 points) What is a drawback to making the hash table big enough so the chances of a collision are extremely small?

A. It makes the hash function less effective.

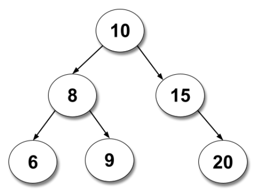
B. It increases the cost of resizing the hash table.

C. It reduces the load factor of the hash table.

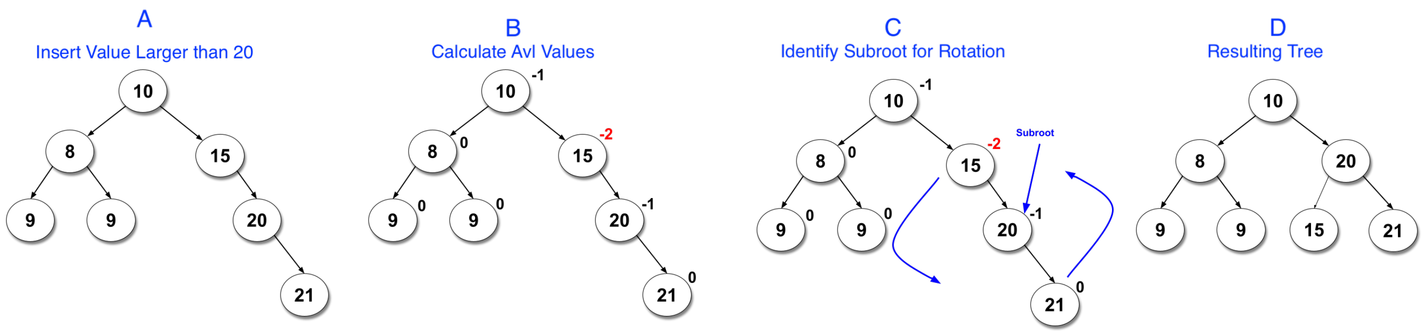
D. It wastes memory resources.

2. SHORT ANSWER

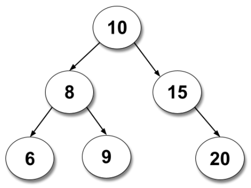
22. (5 points) Given the following AVL Tree. What value could you insert into the AVL Tree that could cause a single rotation after balance factors are calculated? Draw the resulting AVL Tree if you can identify a value that would cause a single rotation.



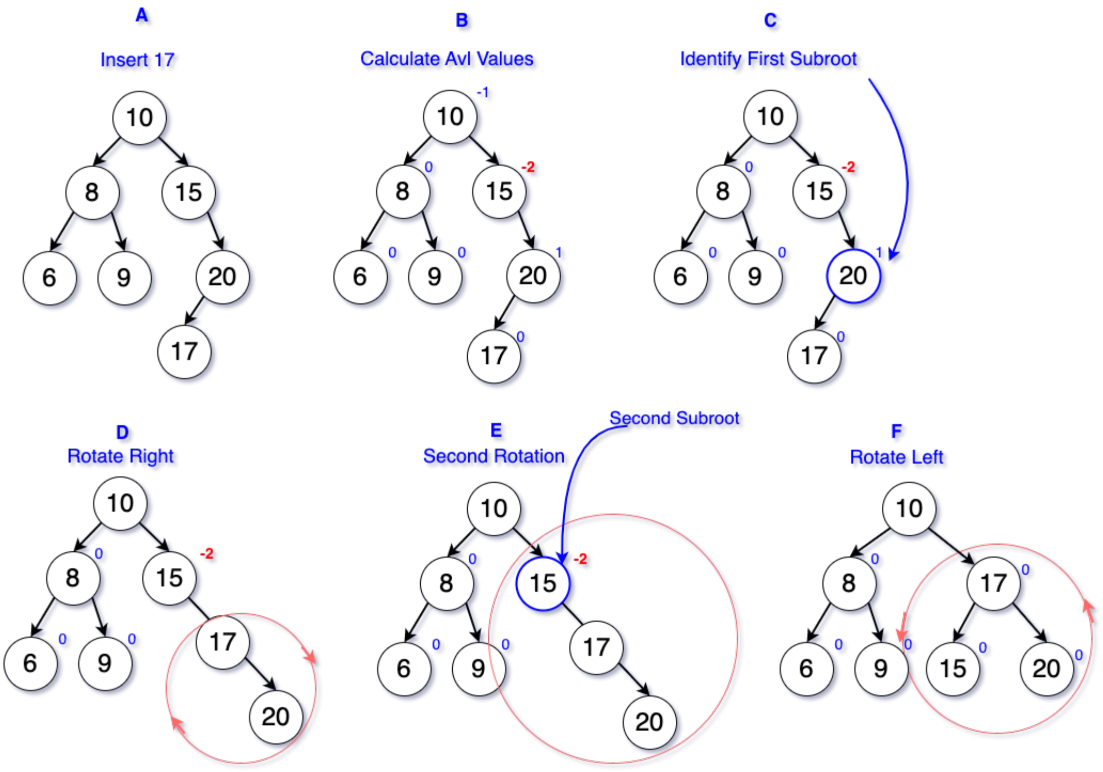
Answer:
- **A:** Choose any value larger than 20. So we insert a value 21.
- **B:** We then calculate avl values and identify the subroot.
- **C:** Then perform the rotation.
- **D:** Final tree.



23. (10 points) Given the following AVL Tree. What value could you insert into the AVL Tree that could cause a double rotation after balance factors are calculated? Draw the resulting AVL Tree if you can identify a value that would cause a double rotation.



Answer:
Any value greater than 15 and less than 20 making the right subtree left heavy.



24. (10 points) Please explain how the worst case efficiency of finding a value in a hash table can degrade to $O(n)$. Explain your answer in terms of both categories of collision resolution: open addressing and chaining.

Answer:
Your answer should include most of the following discussion points:

- bad hash functions lead to lots of collisions
- many collisions leads to either:
  - using chaining you get very long chains
  - using open addressing you get very large clusters
- very small table size effects collisions as well

25. (15 points) Suppose you have the following set of integer keys to insert into a hash table with a table size of 7.
Hash Function: $function\ h(k) = k\ \%\ 7$
Values to be Hashed: $8, 2, 5, 9, 1, 7, 3, 6, 4$

Using three different collision resolution techniques (linear probing, quadratic probing, and chaining), fill out the resulting hash tables after all the keys have been inserted. Show the final state of each table.

Answer:
Linear Probing:
Since the number of values to be hashed was 9, and the table size was 7, the values: 6,4 would be problematic. Since overwriting previously hashed values is not an option, you would have to resize the table. I did not expect you to do this, but mention it in your answer.

[0] => 7
[1] => 8
[2] => 2
[3] => 9
[4] => 1
[5] => 5
[6] => 3

Quadratic Probing:
The probing sequence will not find a location for value 1. Therefore, you cannot really continue to hash the remaining values unless you resize the table.

[0] => None
[1] => 8
[2] => 2
[3] => 9
[4] => None
[5] => 5
[6] => None

Chaining:
Chaining does not have a problem with a load factor > 1. It just creates longer chains. This is good if you don't want to worry about resizing the table. Its bad since are constant time wish of O(1) lookup degrades to longest chain.

[0] => [7]
[1] => [8, 1]
[2] => [2, 9]
[3] => [3]
[4] => [4]
[5] => [5]
[6] => [6]