Student Name: _____ Student ID # _____

OU Academic Integrity Pledge

On my honor I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.

Signature: _____ Date: _____

Notes Regarding this Examination

Open Book(s) You may consult any printed textbooks in your immediate possession during the course of this examination.

Open Notes You may consult any printed notes in your immediate possession during the course of this examination.

No Electronic Devices Permitted You may not use any electronic devices during the course of this examination, including but not limited to calculators, computers, and cellular phones. All electronic devices in the student's possession must be turned off and placed out of sight (for example, in the student's own pocket or backpack) for the duration of the examination.

Violations Copying another's work, or possession of electronic computing or communication devices in the testing area, is cheating and grounds for penalties in accordance with school policies.

Definitions of Time and Space Complexity

Definition of Big O: Let $f(n)$ and $g(n)$ be functions mapping non-negative integers to real numbers. We say that $f(n) \in O(g(n))$ if there is a real number $c > 0$ and a fixed integer $n_0 \geq 1$ such that $f(n) \leq cg(n)$ for every integer $n \geq n_0$.

Definition of Big Ω: Let $f(n)$ and $g(n)$ be functions mapping non-negative integers to real numbers. We say that $f(n) \in \Omega(g(n))$ if there is a real number $c > 0$ and a fixed integer $n_0 \geq 1$ such that $f(n) \geq cg(n)$ for every integer $n \geq n_0$.

Definition of Big Θ: Let $f(n)$ and $g(n)$ be functions mapping non-negative integers to real numbers. We say that $f(n) \in \Theta(g(n))$ if there are real numbers $c, d > 0$ and a fixed integer $n_0 \geq 1$ such that $dg(n) \leq f(n) \leq cg(n)$ for every integer $n \geq n_0$.

Note: Recall that, if an algorithm's Big O complexity puts it in one complexity class, then that algorithm also belongs to every higher Big O complexity class. (For example, if $f(n) \in O(n^5)$, then $f(n) \in O(n^6)$, $f(n) \in O(n^7)$, etc.) For this reason, we are usually interested in an algorithm's *minimum* Big O complexity, that is, the Big O complexity class to which it belongs that provides the lowest ceiling on its performance. Similarly, if an algorithm's Big Ω complexity puts it in one complexity class, then that algorithm also belongs to every lower Big Ω complexity class. (For example, if $f(n) \in \Omega(n^5)$, then $f(n) \in \Omega(n^4)$, $f(n) \in \Omega(n^3)$, etc.) For this reason, we are usually interested in an algorithm's *maximum* Big Ω complexity, that is, the complexity class to which it belongs that provides the highest floor on its performance. If an algorithm's minimum Big O complexity class is the same as its maximum Big Ω complexity class, that complexity class is it's Big Θ complexity class.

Part I. Data Structures Comparisons

1. (2 points) Which data structure has the *smallest minimum Big O* runtime for finding an arbitrary element?
   A. AVL Tree
   B. Sorted Array
   C. Sorted Linked List
   D. Hash Table
   E. A and B

2. (2 points) Which data structure has the *smallest maximum Big Ω* runtime for finding an arbitrary element?
   A. AVL Tree
   B. Sorted Array
   C. Sorted Linked List
   D. Hash Table
   E. A and B

3. (2 points) Which data structure has the *largest minimum Big O* runtime for finding an arbitrary element?
   A. AVL Tree
   B. Sorted Array
   C. Sorted Linked List
   D. Hash Table
   E. C and D

4. (2 points) Which data structure has the *largest maximum Big Ω* runtime for finding an arbitrary element?
   A. AVL Tree
   B. Sorted Array
   C. Sorted Linked List
   D. Hash Table
   E. C and D

5. (2 points) Which data structure has the same minimum Big O runtime as a Minimum Heap for *finding and removing the smallest element*?
   A. AVL Tree
   B. Sorted Array
   C. Sorted Linked List
   D. Hash Table
   E. C and D

6. (2 points) Which data structure has the same minimum Big O runtime as a Minimum Heap for *inserting an arbitrary element*?
   A. AVL Tree
   B. Sorted Array
   C. Sorted Linked List
   D. Hash Table
   E. A and B

Part II. Trees

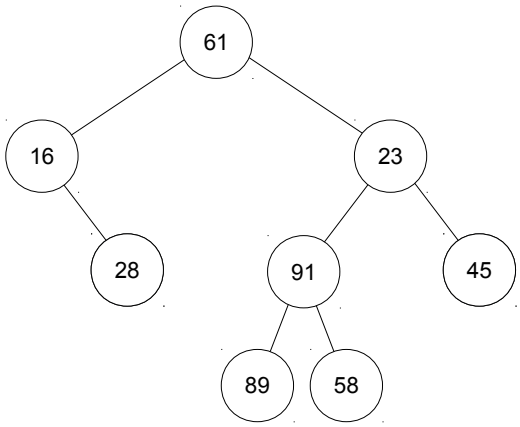
7. (2 points) Which of the following has a runtime of $\Omega(\log n)$ for *finding* an arbitrary element?
   A. AVL Tree
   B. Red-Black Tree
   C. 2-3 Tree
   D. All of the Above
   E. None of the Above

8. (2 points) Which of the following has a runtime of $\Omega(\log n)$ for *adding* an arbitrary element?
   A. AVL Tree
   B. Red-Black Tree
   C. 2-3 Tree
   D. All of the Above
   E. None of the Above

9. (2 points) Which of the following has a runtime of $\Omega(\log n)$ for *removing* an arbitrary element?
   A. AVL Tree
   B. Red-Black Tree
   C. 2-3 Tree
   D. All of the Above
   E. None of the Above

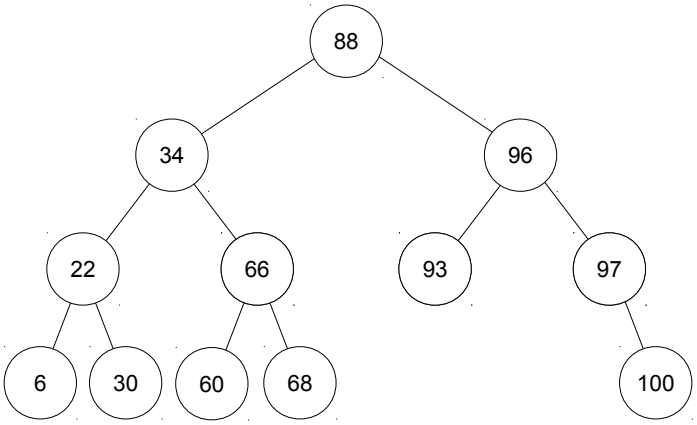
10. (2 points) Which of the following has a *height* of $\Omega(\log n)$?
    A. AVL Tree
    B. Red-Black Tree
    C. 2-3 Tree
    D. All of the Above
    E. None of the Above

Short Answer Question 1: Tree Traversal (5 points)



Show the postorder traversal of the above binary tree.

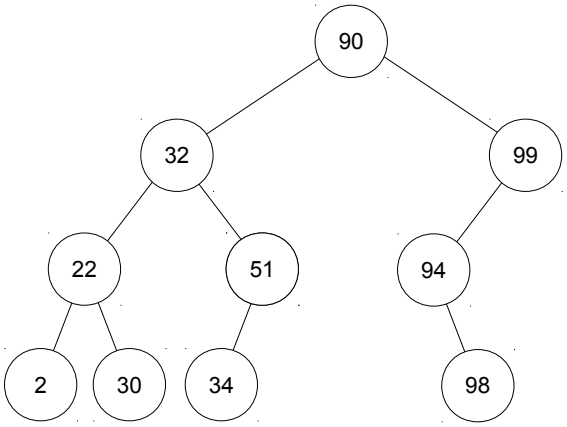
Short Answer Question 2: Binary Search Trees (10 points)



A. **Explain** how many keys need to be compared to perform a search for the key 66 in the tree above.

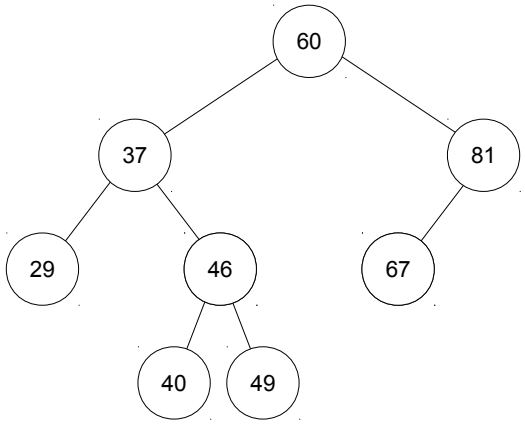
B. **Explain** how many keys need to be compared to perform a search for the key 95 in the tree above.

Short Answer Question 3: Self-Modifying Search Trees (10 points)



Explain the result of performing a zigzag operation on the node with key 99 in the tree above.

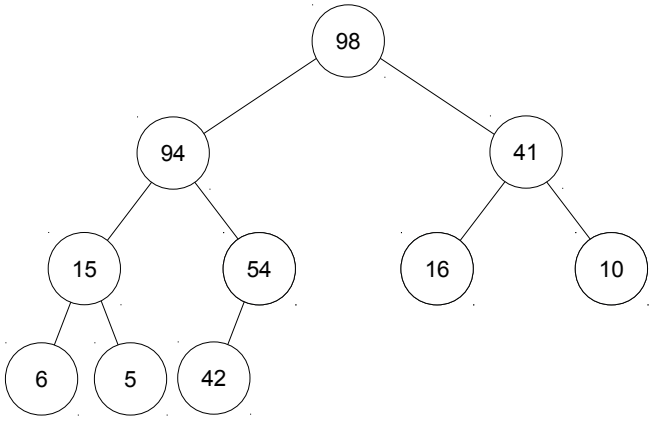
Short Answer Question 4: AVL Trees (15 points)



Explain the result of adding the node with key 48 to the above tree.

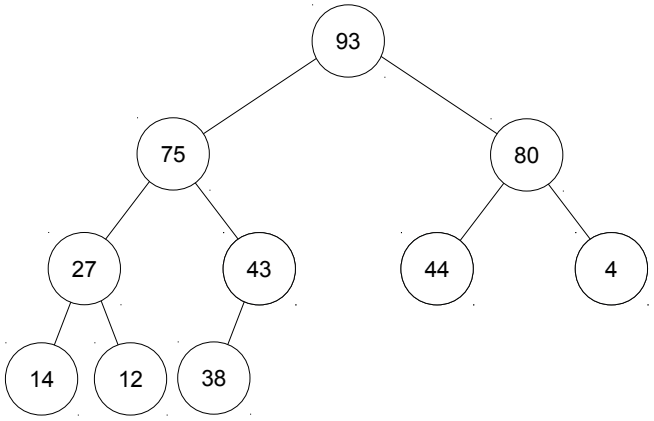
Short Answer Question 5: Priority Structures (10 points)

Show the steps required to add the key 100 to the following Max Heap.



Short Answer Question 6: Priority Structures (10 points)

Show the steps required to remove the next item from the following Max Heap.



Short Answer Question 7: Ethics (20 points)

"Put it in the cloud. Put it in the cloud. Put it in the cloud." The words echoed in Mamoru's head. It seemed that was all he heard these days. Well, fine, he'd try to put his files in the cloud. But the user interface for his school-provided cloud service was awful. "Who developed this UI anyway?" Everything was set up to back up his files but he didn't want to have backups in the cloud, he wanted to copy his files to the cloud and delete them off his laptop. After all, the whole reason he was doing this was because his laptop's hard drive was almost completely full.

Still, what could he do? He desperately needed to free up some space on his local drive just to complete the huge final projects in his classes and finish out the semester, he was too tired and too focused on completing his last projects to think clearly about what files could be safely deleted permanently, and all of his free cloud storage was full except, of course, the space he had through his school. The space with the terrible UI. "Aaaaaargh!"

Fine. Fine. Reasoning about the UI didn't work. Searching the web for answers didn't work. Mamoru decided to just go through the menus looking for something, anything, that could make it work. "The functionality has to be here!" He told himself. "It was here in the old version—the one all the advice on the web targets. They must have simply moved it somewhere else. But where?"

Mamoru clicked and dragged and double clicked and scrolled and clicked some more but to no avail. Then, suddenly, a whole new folder of files appeared in the cloud—files with names he didn't recognize. Lots of them. And lots of folders too, also with names he didn't recognize. "What in the world?"

He clicked on one of the folders. More unrecognized files. More unrecognized folders. He dived into one of those folders. Even more files and even more folders. "Weird." He clicked on one of the files. It opened. He hadn't recognized the file type but it opened as a spreadsheet. Names. Numbers. Odd little three-letter codes. It wasn't a grade file. He wasn't sure what it was. He clicked on another file. It turned out to be another spreadsheet. This one appeared to be financial records of some kind—lots of dollar signs and calculations.

Mamoru was puzzled—curious but also exhausted and defeated. It was late. Maybe he'd have better luck in the morning. He closed his laptop and headed to bed.

A. Find at least one computer crimes law that is relevant to this scenario. List the name of the law and *explain* why you think it is relevant.

B. Say whether you think Mamoru abided by (that is, followed) the law you listed and *explain* how you came to that conclusion.

C. Give one likely motivation for Mamoru's action and *explain* how you concluded that was a likely motivation.

D. List one ethical-decision-making problem (interfering factor) that is likely to have contributed to at least one of Mamoru's decisions and *explain* how you concluded that was a likely problem.

E. List one ethical-decision-making strategy that Mamoru could employ to improve his ethical decision making and *explain* how he might employ that strategy in this situation.