Bonus Exercise: Complexity

Example: Strange Program

Input An integer n

Output ???

Task Calculate the UT this strange program takes with input n

```
int goomy(int n){
 goomyFactor = 0;
 for(i=0; i< n; i++){
  if(n mod 2 = 0){
   goomyFactor = goomyFactor * 2;
  }
  else{
   goomyFactor = goomyFactor - 1;
  }
 }
 return goomyFactor;
}
```

- Per loop: 4 (1 for loop, 1 for if, 1 for operation and assignment (in if or else))
- Loop: $4 \cdot n$
- Outside of loop: 2 (initialize variable and return)
- Function goomy(n): $4n + 2$

Bonus Exercise: Complexity

Example: Another strange program

Input An integer n

Output ???

Task Calculate the UT this strange program takes with input n

```
int descent(int n){
 loot = 0;
 for(i=0; i< n; i++){
   loot = loot + n mod 7;
 }
 if(n>1) {
  return loot + descent(n -1);
 }
 else {
  return loot + 1;
 }
}
```

- Loop: $4 \cdot n$
- Outside of loop: 4
- But: function calls itself
- descent(n):
  $n \cdot 4 + 4 + descent(n-1) =$
  $4(n+1) + descent(n-1)$
- $\sum_{i=1}^{n} 4 \cdot (i+1) = 4 \cdot \sum_{i=1}^{n}(i+1) =$
  $4 \cdot (\sum_{i=1}^{n} i + \sum_{i=1}^{n} 1) =$
  $4 \cdot ((n+1) \cdot n/2 + n) =$
  $4 \cdot (n^2/2 + n/2 + n) = 2n^2 + 6n$

Bonus Exercise: Complexity

Disprove by contradiction: $n \cdot \sqrt{n} \in \mathcal{O}(n)$

$n \cdot \sqrt{n} \notin \mathcal{O}(n)$. Consider the definition of $\mathcal{O}(n)$:
$\exists k \in \mathbb{N} \quad \exists c \in \mathbb{R}^{\geq 0} \quad \forall n > k : n \cdot \sqrt{n} \leq c \cdot n$
The comparison can be simplified to $\sqrt{n} \leq c$.

Choose $n = (c + k + 1)^2$. Then it holds $n > k$.

If $n \cdot \sqrt{n} \in \mathcal{O}(n)$, then:

$\sqrt{(c + k + 1)^2} \leq c$
$c + k + 1 \leq c$
$k \leq -1$, but we defined $k \in \mathbb{N}$, which is a contradiction.

Therefore $n \cdot \sqrt{n} \notin \mathcal{O}(n)$. QED.

Bonus Exercise: Complexity

Show or disprove: $sin(n) \in \mathcal{O}(1)$

$sin(n) \in \mathcal{O}(1)$. Consider $c = 1$, $k = 1$. Then it holds for all $n \in \mathbb{N}, n > k : sin(n) \leq 1 = c * 1$ and therefore the claim by definition of $\mathcal{O}$.

Show or disprove: $e^x \in \mathcal{O}(e^{2x})$

$e^x \in \mathcal{O}(e^{2x})$: $\lim_{n\to\infty} \frac{e^x}{e^{2x}} = \lim_{n\to\infty} \frac{e^x}{e^x * e^x} = \lim_{n\to\infty} \frac{1}{e^x} = 0 \in \mathbb{R}$

Show or disprove with l'Hopital: $n * \ln(n) \in \mathcal{O}(n^2)$

$n * \ln(n) \in \mathcal{O}(n^2)$ with l'Hopital:
$\lim_{n\to\infty} \frac{n*\ln(n)}{n^2} = \lim_{n\to\infty} \frac{ln(n)}{n} = \lim_{n\to\infty} \frac{\frac{1}{n}}{1} = \lim_{n\to\infty} \frac{1}{n} = 0 \in \mathbb{R}$

Show or disprove: $\sqrt{n} * \ln(n) \in \mathcal{O}(n)$

$\sqrt{n} * \ln(n) \in \mathcal{O}(n)$ with l'Hopital: $\lim_{n\to\infty} \frac{\sqrt{n}*\ln(n)}{n} = \lim_{n\to\infty} \frac{\ln(n)}{\sqrt{n}} =$
$\lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{2\sqrt{n}}{n} = \lim_{n\to\infty} \frac{2}{\sqrt{n}} = 0 \in \mathbb{R}$

Bonus Exercise: Complexity

Show or disprove: $x \log_2 x \in O(x^2)$

Assumption: $x \log_2 x \in O(x^2)$

Use limit rule: $\lim_{x\to\infty} \frac{x \log_2 x}{x^2} \in \mathbb{R}^{\geq 0}$.

$$\begin{aligned}
\lim_{x\to\infty} \frac{x \log_2 x}{x^2} &= \lim_{x\to\infty} \frac{\log_2 x}{x} \text{ (reduce } x\text{)} \\
&= \lim_{x\to\infty} \frac{\frac{1}{\ln(2)x}}{1} \text{ (l'Hôpital)} \\
&= \lim_{x\to\infty} \frac{1}{\ln(2)x} \text{ (calculation)} \\
&= 0
\end{aligned}$$

Therefore: The limit exists and is in $\mathbb{R}^{\geq 0}$. The assumption is proven.

Bonus Exercise: Logarithms

Calculate with mental arithmetics: $\log_4(32)$

$\log_4(32) = \mathit{log}_4(4 \cdot 4 \cdot 2) = \log_4(4) + \log_4(4) + \log_4(2)$
We know: $\log_x(x) = x$ and $2 = \sqrt{4} = 4^{\frac{1}{2}}$
Therefore: $\log_4(4) + \log_4(4) + \log_4(2) = 1 + 1 + \frac{1}{2} = 2.5$
Verify using exponent: $4^{2.5} = 32 = 2^{2 \cdot 2.5} = 2^5$

Calculate as a function of x: $\log_b(x^0)$
$\log_b(x^0) = \log_b 1 = 0$, as for all bases b: $\log_b 1 = 0$

Calculate as a function of x: $\log_x(\sqrt{x})$
$\log_x(\sqrt{x}) = \log_x x^{\frac{1}{2}} = \frac{1}{2}$

Calculate as a function of x: $\log_x(\frac{1}{x})$
$\log_x(\frac{1}{x}) = \log_x x^{-1} = -1$

Bonus Exercise: Logarithms and complexity

Prove or disprove **without** using L'Hôpital's rule:
$\forall a > 1 : \log_a(e^n) \in \mathcal{O}(n)$

We know we can change logarithm base like this: $\log_a x = \frac{\log_b x}{\log_b a}$

$\log_a(e^n) = \frac{\ln(e^n)}{\ln a} = \frac{1}{\ln a} \cdot \ln(e^n) = \frac{1}{\ln a} \cdot n = c \cdot n \in \mathcal{O}(n)$

Prove or disprove **without** using L'Hôpital's rule: $\log(n^2) \in \mathcal{O}(\log n)$

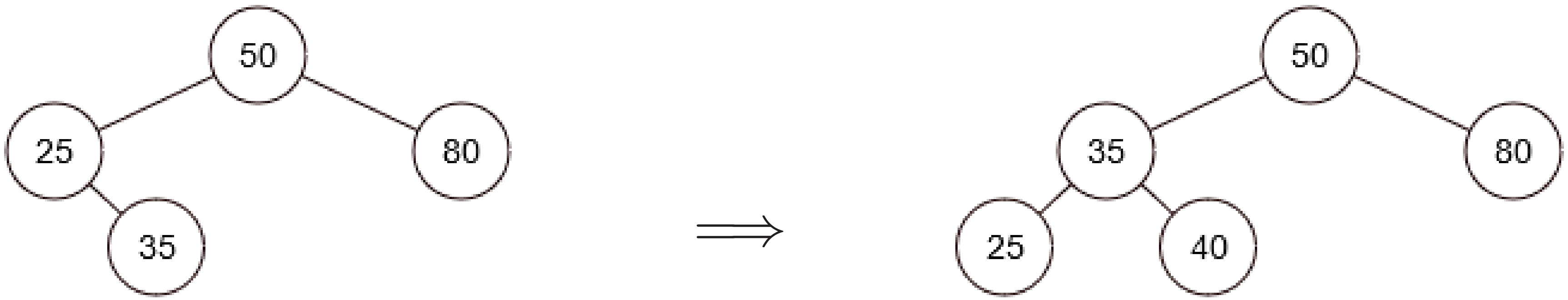
$\log(n^2) = \log(n \cdot n) = \log(n) + \log(n) = 2 \cdot \log n = c \cdot \log n \in \mathcal{O}(\log n$

Prove or disprove **without** using L'Hôpital's rule: $\sqrt{2}^{\log_2 n} \in \mathcal{O}(\sqrt{n})$

$\sqrt{2}^{\log_2 n} = 2^{\frac{1}{2} \cdot \log_2 n} = 2^{\cdot \log_2(n^{\frac{1}{2}})} = n^{\frac{1}{2}} = \sqrt{n} \in \mathcal{O}(\sqrt{n})$

Bonus Exercise: AVL trees

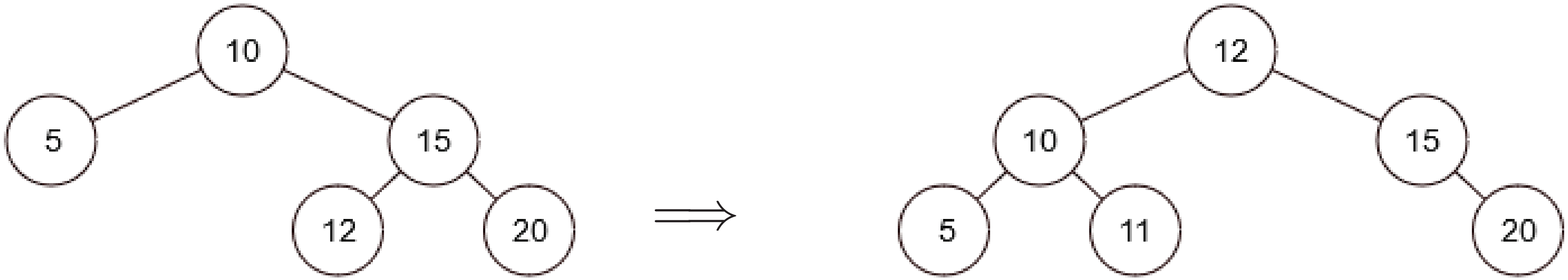
Insert **40** into the AVL tree. Write down all necessary rotations.



Right rotation with pivot **35** and root **25**.

Bonus Exercise: AVL trees

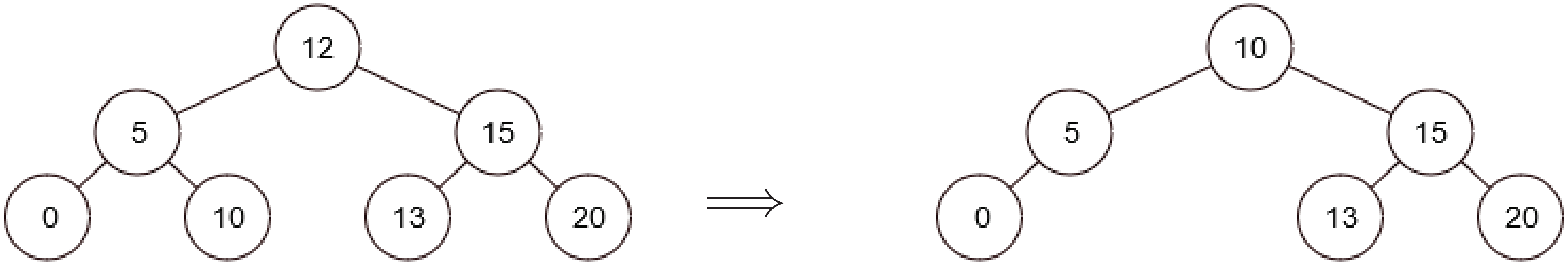
Insert **11** into the AVL tree. Write down all necessary rotations.



Right-Left rotation with pivot **15** and root **10**. First rotate right with pivot **15** as root. Then rotate left with root **10** and NEW pivot **12**.

Bonus Exercise: AVL trees

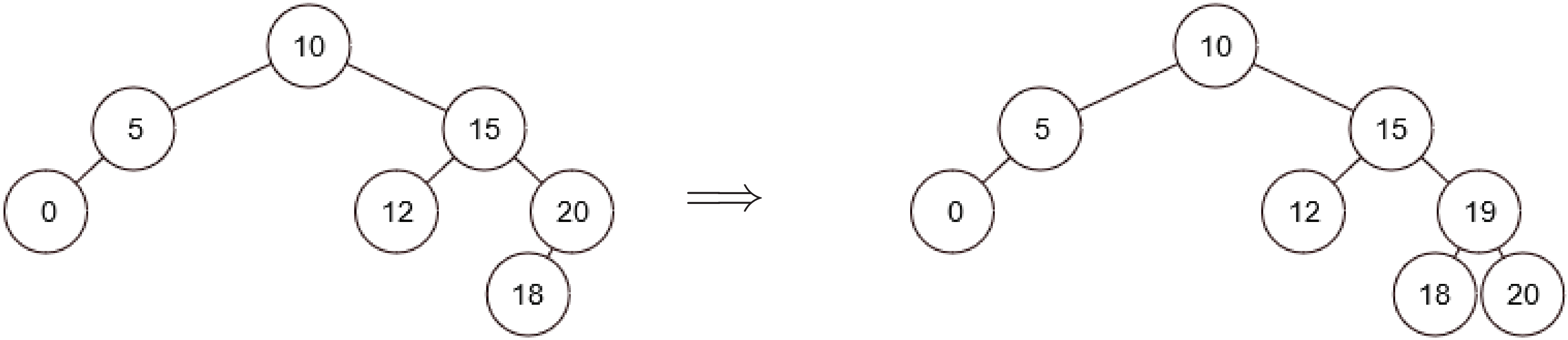
Delete **12** from the AVL tree.Write down all necessary rotations.



No rotations are necessary.

Bonus Exercise: AVL trees

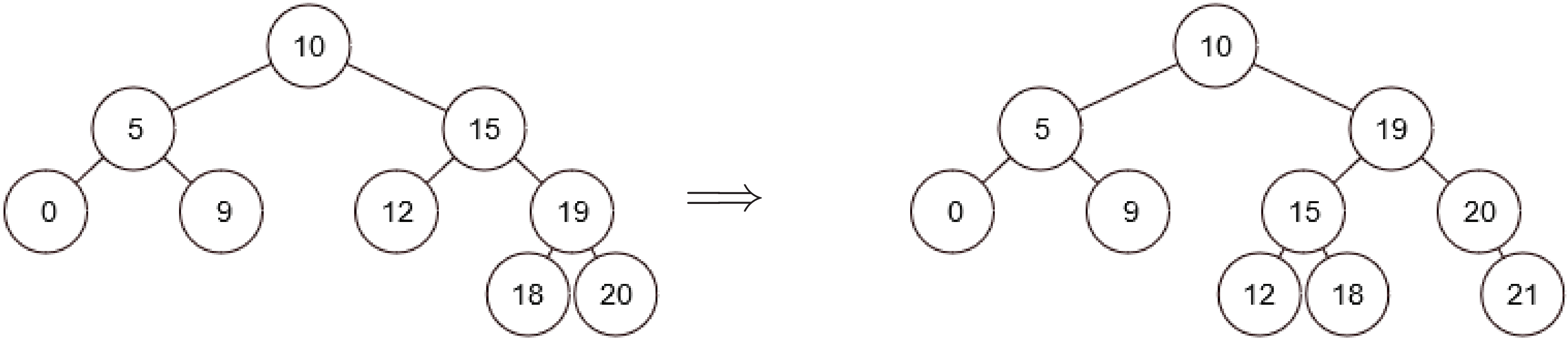
Insert **19** into the AVL tree. Write down all necessary rotations.



Left-Right rotation with pivot **18** and root **20**. First rotate left with pivot **18** as root. Then rotate right with root **20** and NEW pivot **19**.

Bonus Exercise: AVL trees

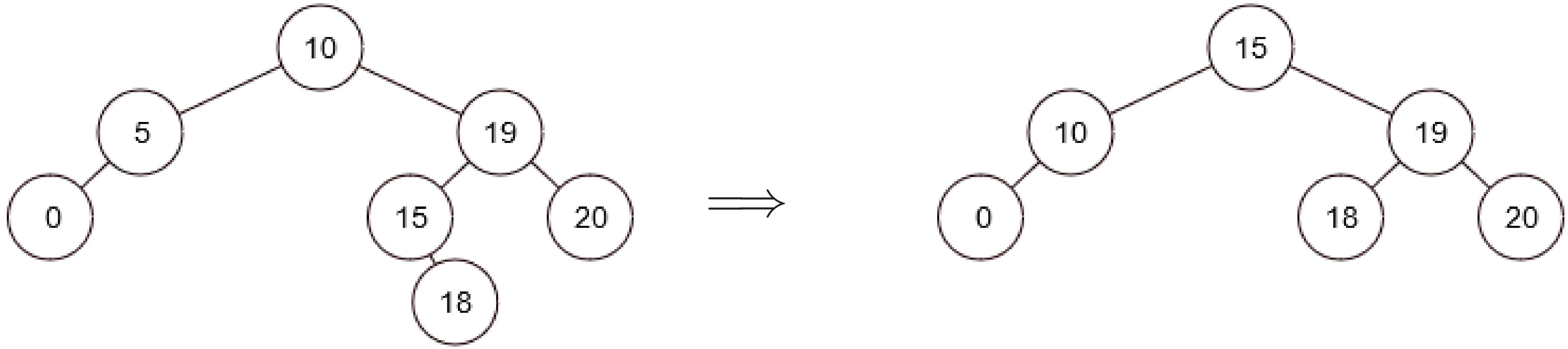
Insert **21** into the AVL tree. Write down all necessary rotations.



Left rotation with pivot **19** and root **15**.

Bonus Exercise: AVL trees

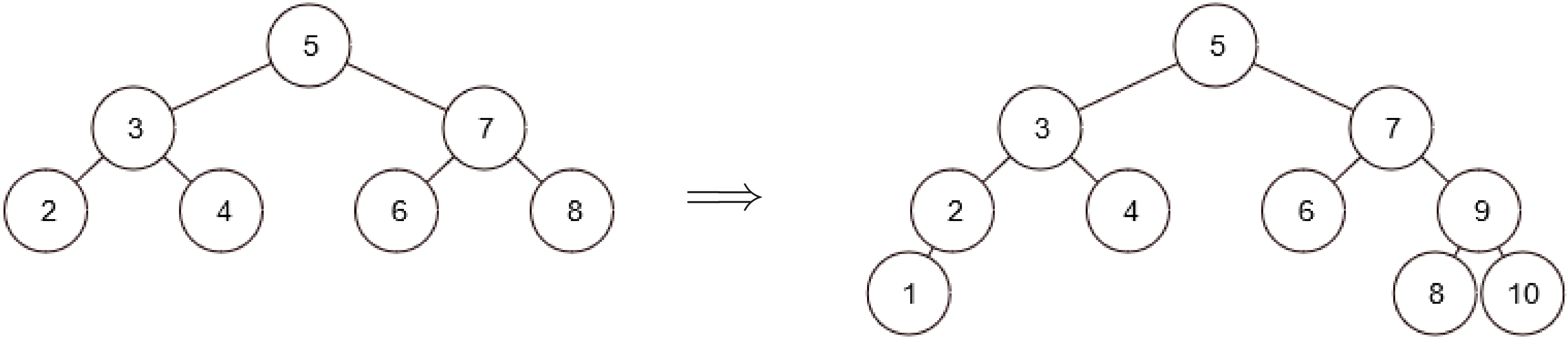
Delete **5** from the AVL tree. Write down all necessary rotations.



Right-Left rotation with pivot **19** and root **10**. First rotate right with pivot **19** as root. Then rotate left with root **10** and NEW pivot **15**.

Bonus Exercise: AVL trees

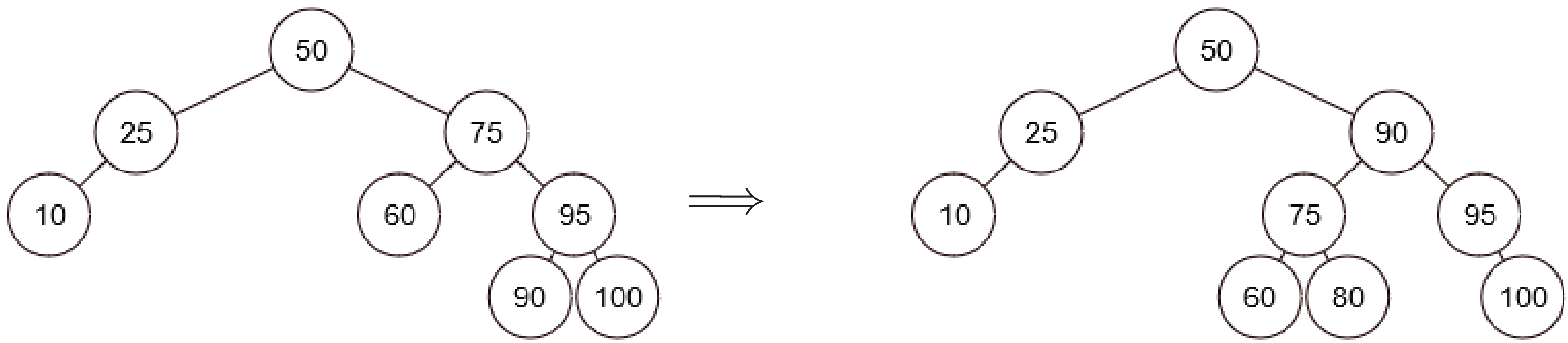
Insert **1**, then **9**, then **10** into the AVL tree. Write down all necessary rotations.



No rotations for 1.
No rotations for 9.
Left rotation with pivot **9** and root **8** for 10.

Bonus Exercise: AVL trees

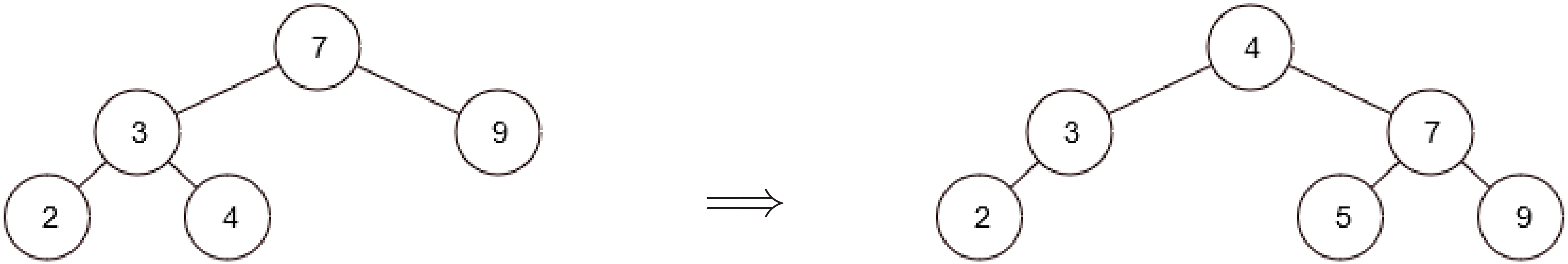
Insert **80** into the AVL tree. Write down all necessary rotations.



Right-Left rotation with pivot **95** and root **75**. First rotate right with pivot **95** as root. Then rotate left with root **75** and NEW pivot **90**.

Bonus Exercise: AVL trees

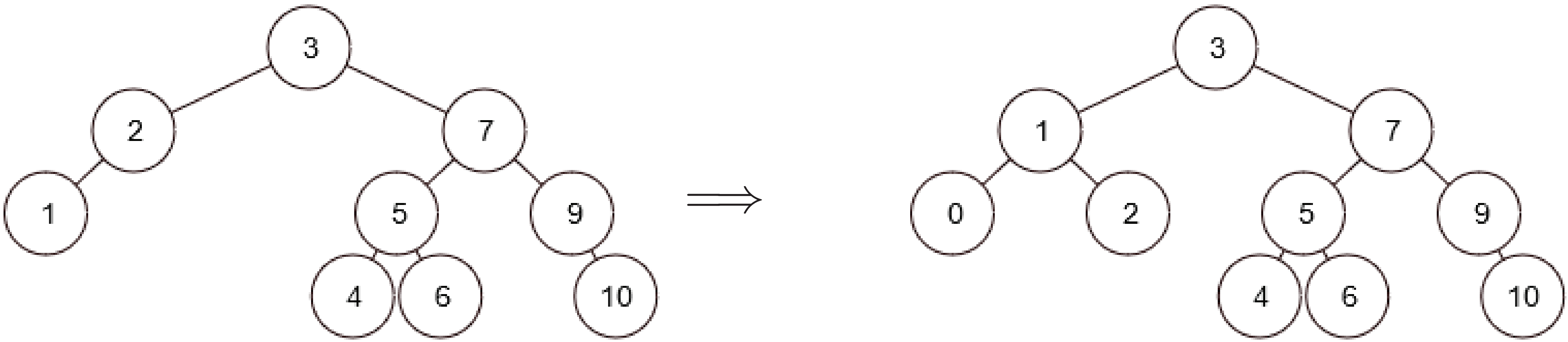
Insert **5** into the AVL tree. Write down all necessary rotations.



Left-Right rotation with pivot **3** and root **7**. First rotate left with pivot **3** as root. Then rotate right with root **7** and NEW pivot **4**.

Bonus Exercise: AVL trees

Insert **0** into the AVL tree. Write down all necessary rotations.



Right rotation with pivot **1** and root **2**.

Bonus Exercise: Quicksort

Sort the following array with quicksort. Use the last element as a pivot. Show the array after each recursion step. Highlight the pivot element. You can skip steps with only one element.

2	1	7	9	4	6	3	8	5

Choose 5 as pivot and partition from index 0 to 8:

2	1	4	3	5	6	9	8	7

Recursively quicksort for index 0 to 3 (pivot 3) and index 5 to 8 (pivot 7):

2	1	3	4	5	6	7	8	9

Recursively quicksort for index 0-1, 3-3, 5-5, 7-8:

1	2	3	4	5	6	7	8	9

Some more recursive calls with size 1 may follow, but they do not change the array.

Bonus Exercise: Hashing

Insert the following values in a hash table of size 10:
3, 12, 77, 13, 96, 55, 50
Use the digit sum as a hashing function (e.g. 123 $\rightarrow$ 6, 71 $\rightarrow$ 8). Handle collisions by re-hashing using the first digit (e.g. 123 $\rightarrow$ 1, 71 $\rightarrow$ 7).

Mark all fields in which collisions occur. Notice any problems?

0						55	55 C 2,4,6,...
1							
2					96	96	96
3	3	3 C	3	3	3 C3	3	3
4			77	77 C1	77 C2	77	77
5		21	21	21 C2	21 C1	21	21 C1,3,5,...
6				13	13	13	13
7							
8							
9							

It is impossible to insert 55, as the rehashing jumps between index 0 and 5.

This is why you use prime numbers for table size.

Good luck in the exam!