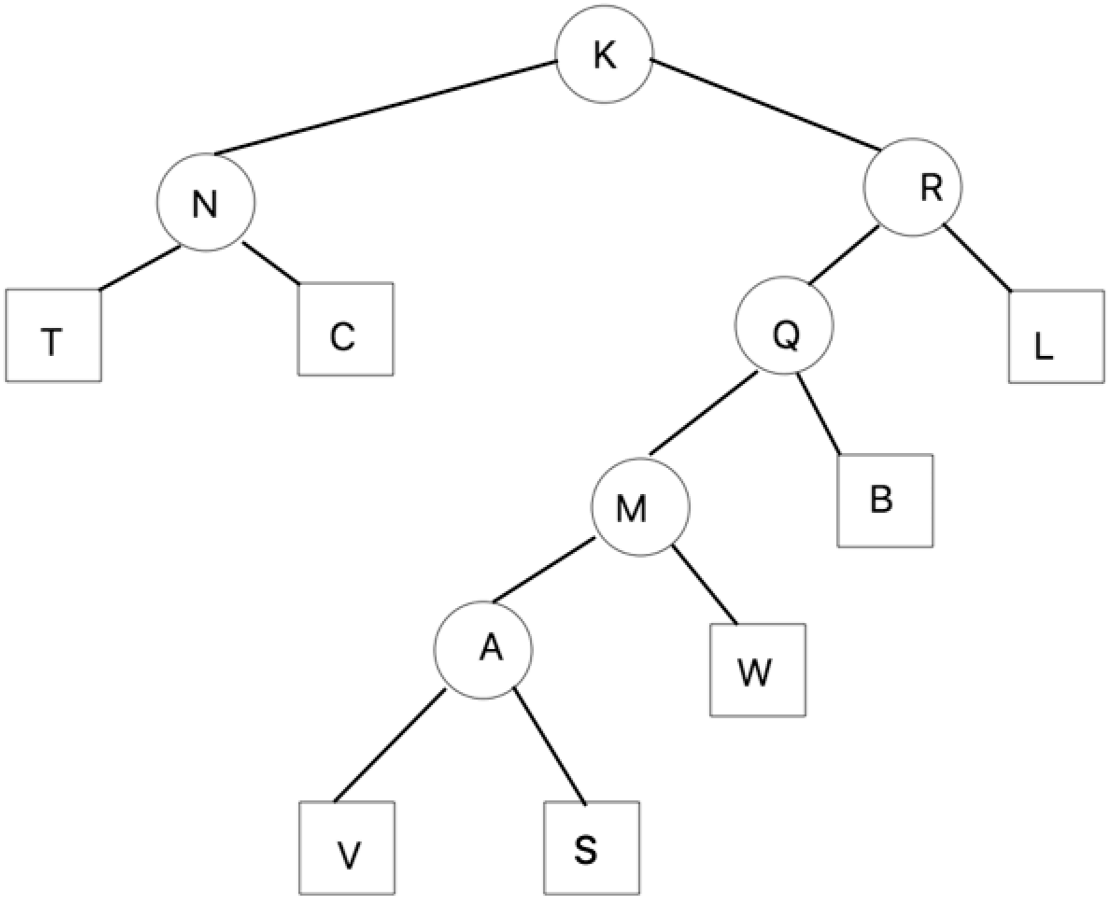
Noemi Lemonnier
Student ID #40001085

## Question 1

**A)**



**B)**
To place all elements, I followed the rule: Left 2n +1 and Right 2n + 2.

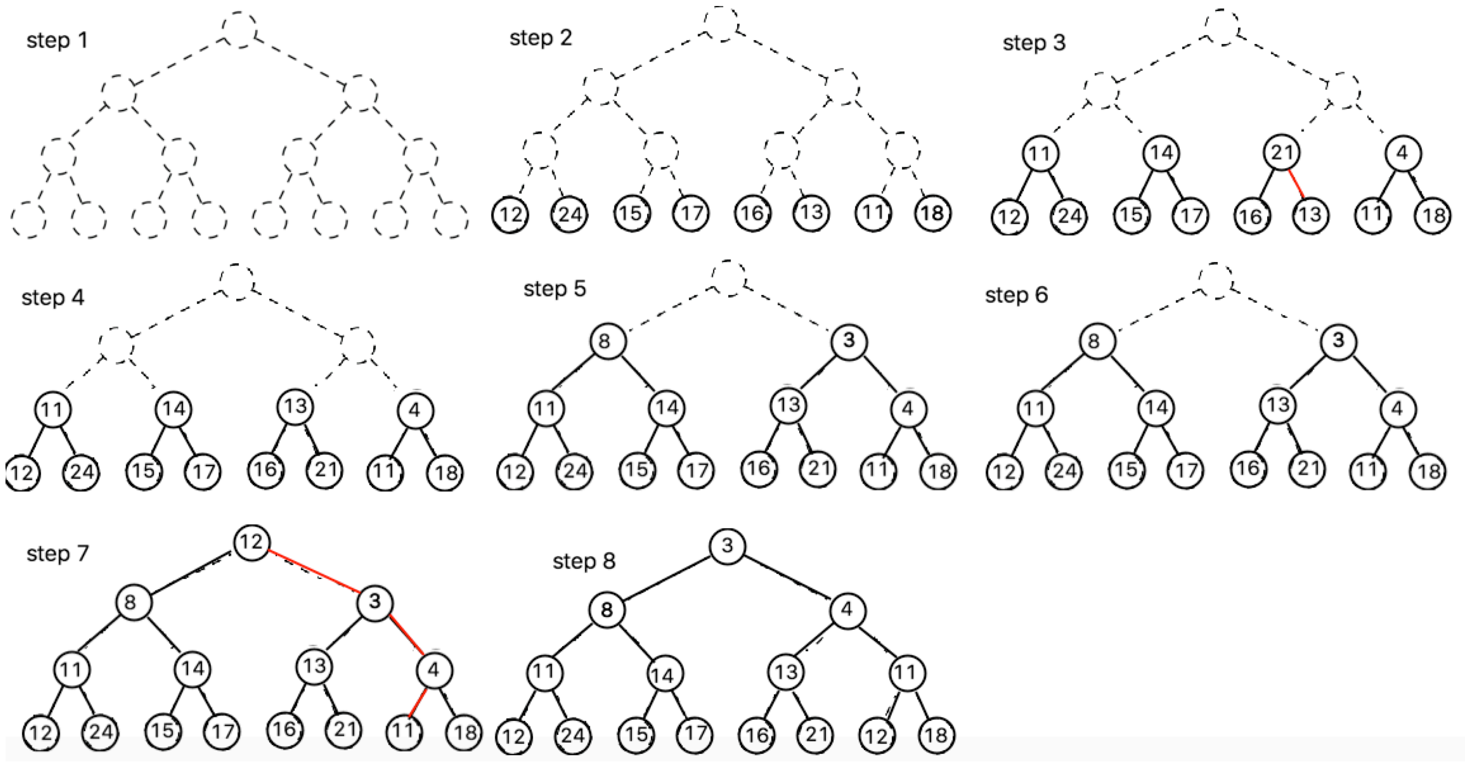
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | N | R | T | C | Q | L | | | | | M | B | | | | | | | | | | | A | W | | | | | | | | | | | | | | | | | | | | | | | V | S |

The root, K, is at index 0. N=2 x 0 + 1=1. R=2x0+2=2. T is left child of N =2x1+1=3 and C is right child of N=2x1+2=4. Q is left child of R=2x2+1=5 and L is right child of R=2x2+2=6. M is left child of Q=2x5+1=11 and B is right child of Q=2x5+2=12. A is left child of M=2x11+1=23 and W is right child of M=2x11+2=24. V is left child of A=2x23+1=47 and S is right child of A=2x23+2= 48.

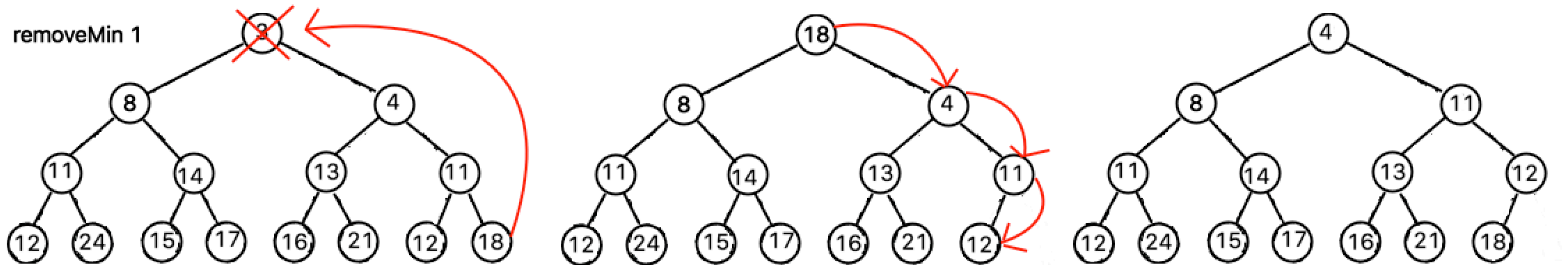
## Question 2

**A)**

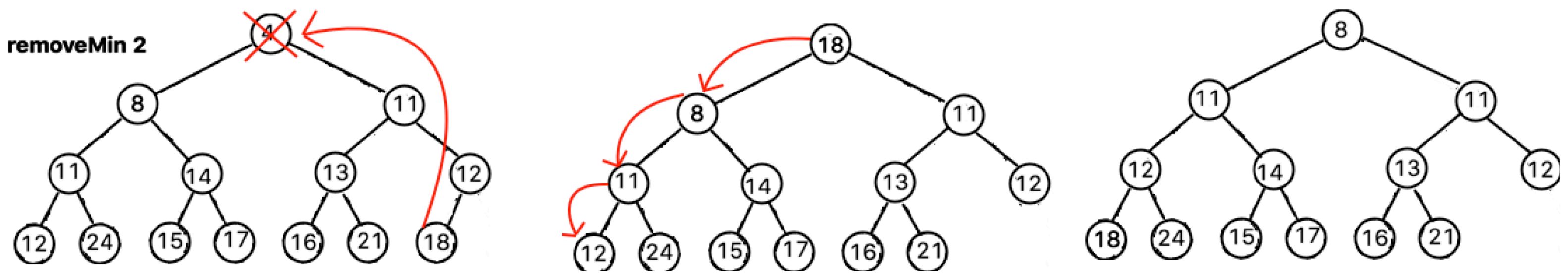
***Step 1***: find the amount of value to put at the bottom of the tree. formula: (n+1)/2
***Step 2***: place the first 8 elements at the bottom
**Step 3**:  place the next 4 elements on top
**Step 4**: make sure the heap order is respected and do the modifications
***Step 5:*** place the next 2 elements
***Step 6***: make sure the heap order is respected and do the modifications
***Step 7***: place the last element
***Step 8***: make sure the heap order is respected and do the modifications



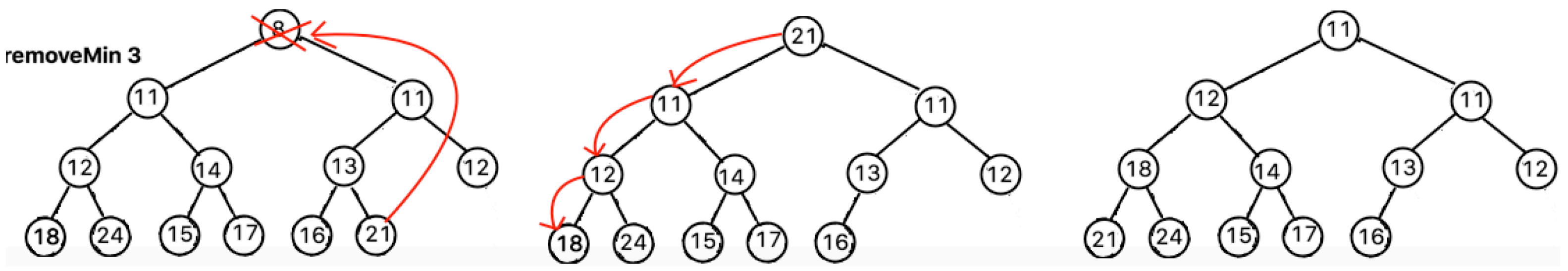
Now performing the **removeMin**:



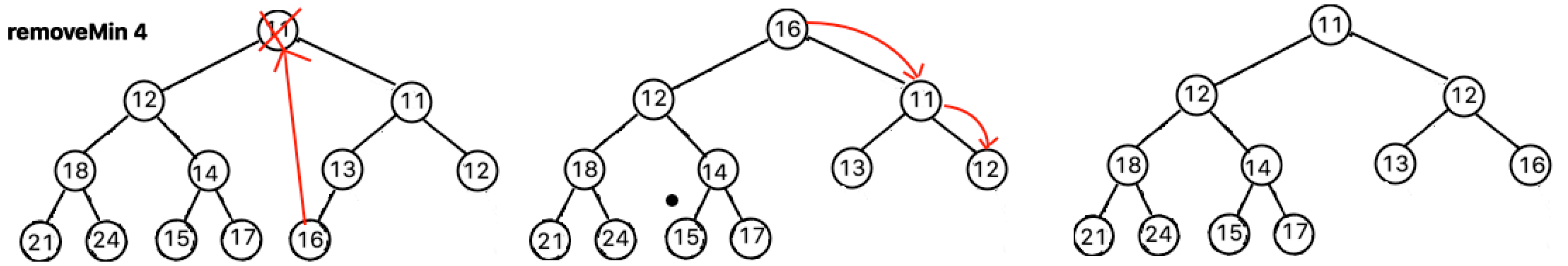
**removeMin 2**



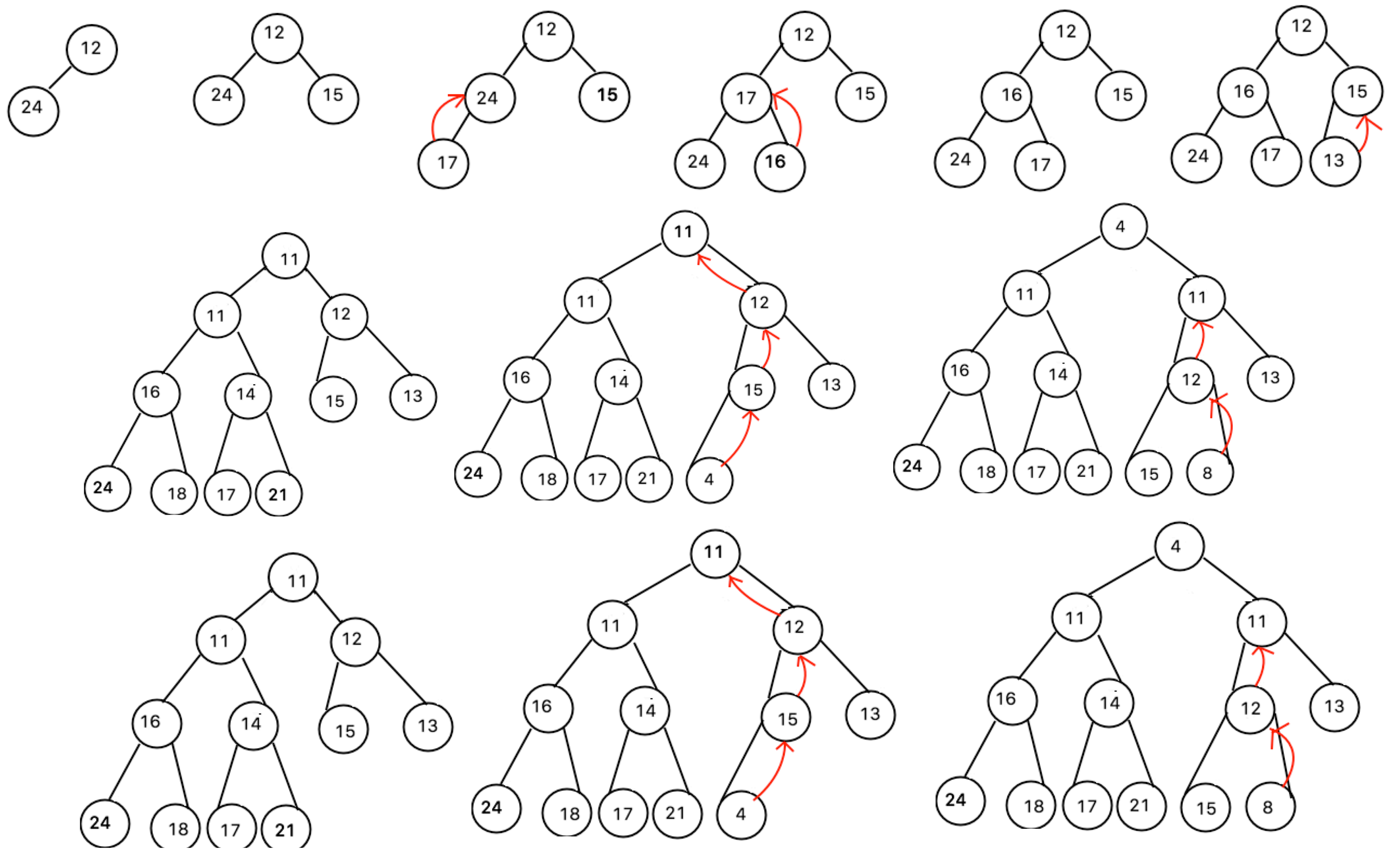
**removeMin 3**

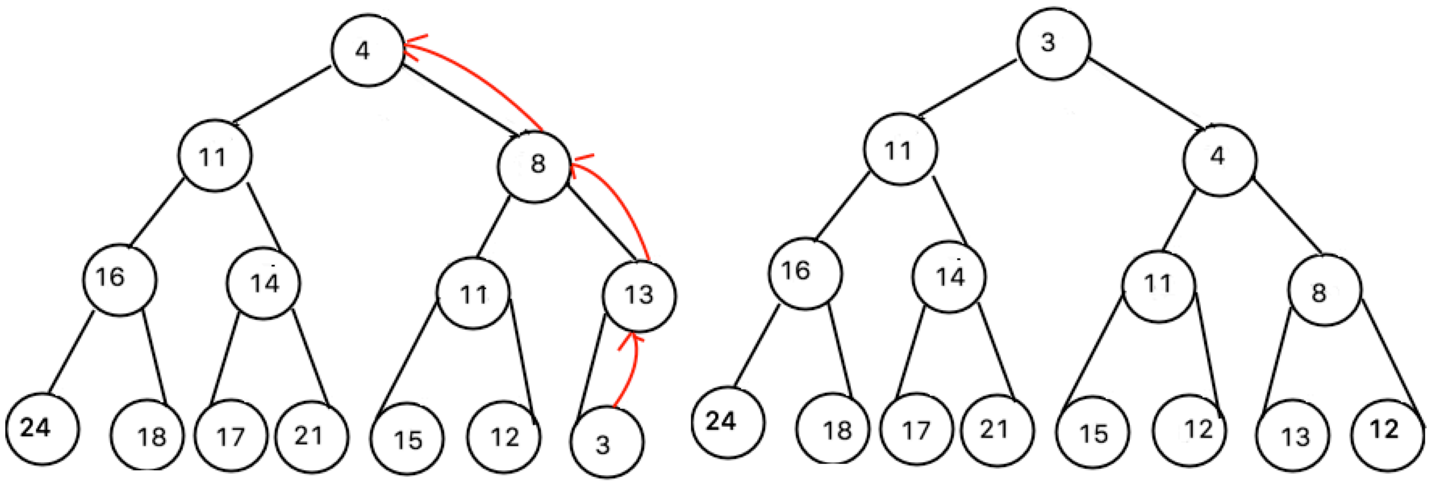


**removeMin 4**



**B)**





## Question 3

```
depthCalculator(Tree T)
        **Input**: Tree T which has n number of nodes
        **Output**: The depth of all nodes of tree T

        Node temp <- T.root()
        temp.depth() == 0
        helperDepth(T, temp)

helperDepth(Tree T, Node temp)
        **Input**: Tree T and Node temp from method depth Calculator
        **for** (i <- all children of temp)
                i.setDepth <- i.getParent.getDepth() +1
                helperDepth(T, i)
```

**A)** The time complexity of the algorithm is O(n) because we need to visit each node once.

**B)** I believe the best time complexity for this algorithm is O(n) because in the best case, the algorithm will need to access once every node to get their depth and do the sum of all nodes' depths.