Test problems.

1. The decision tree below is a decision tree resulting from running Mergesort on a lift with 3 elements [a1,a2,a3].  \textbf{true}

2. The tree below cannot be a decision tree of any comparison-based algorithm after adding all required labelling.  \textbf{true}

3. A decision tree of any comparison-based algorithm has to have all levels full except for the last one.  \textbf{false}

4. The binary tree below is a binary heap:  \textbf{false}
5. Let \([10,9,8,8,7,6,5,3,2,1]\) be an array representation of the binary heap. After removing the element of the highest priority you will get the array \([9,8,3,7,6,5,1,2]\). False. Any answer will be worth 1 point.

6. The following list \([1,3,1,4,5,2]\) achieves the maximum number of swaps and comparisons possible for a list with 6 elements while recursively heapifying it to construct the heap. Note, that you count only comparisons of two elements of the lists and swaps of two elements of the list in this question. True

7. The following tree can be seen at a step of Heapsort algorithm while removing vertices of the highest priority: false

```
        9
       /|
      / |
     7 8
    /   |
   6    5
```

8. The sublist \([12,14]\) will be considered while looking for the 5th order statistic on the list \([5,3,9,10,15,14,20,12]\) using Quickselect algorithm. True

9. 5 will be chosen as the pivot at some step of Quickselect algorithm to find the 3rd order statistic of the list \([17,9,6,5,10,22,14,5]\). True

10. 10 is the 4th order statistic of the list \([14,1,11,2,3,10,7]\) False

11. Selection sort achieves the maximum possible number of comparisons on the list \([1,2,3,4,5,6,7,8,9]\). Note, that you count only comparisons of two elements with at least one of them being an element of a list. True

12. List \([2,5,1,1,7,8,10]\) can be seen at some step of Selectionsort algorithm while sorting \([10,2,5,7,1,8,1]\). False

13. Insertion sort does larger number of elementary operations than Selection sort while sorting the list \([14,6,9,10]\). An elementary operation here is a comparison of two elements with at least one of them being an element of the list. False

14. Lists \([10,15,11,17,16]\), \([10,11,12,9,13]\) and \([14,16,18,20,17]\) take the same number of comparisons while sorting by Insertion sort. You count comparisons only between two elements of a list. True

15. Insertion sort achieves the minimum possible number of comparisons on the list \([1,2,3,4,5,6]\). You count comparisons only between two elements of a list. True

16. List \([23,34,56,2]\) can be seen at some step of Insertionsort algorithm while sorting \([34,23,2,56]\). False

17. Mergesort achieves the minimum possible number of comparisons on the list \([7,6,4,2,3,1]\). You count comparisons only between two elements of a list. True

18. The sublist \([45,2,5]\) will be considered while running Mergesort algorithm on the list \([8,23,45,2,5,7,3,6,9]\). False

19. Mergesort achieves the maximum possible number of comparisons on the list \([5,1,7,3,4,2,6]\). You count comparisons only between two elements of a list. True
20. Suppose you need to order the list \([1_a, 3, 6, 8, 10, 2, 5, 18, 1_b, 4, 19, 23]\), where \(1_a, 1_b\) are two copies of 1. After running mergesort on this list you can get the following list:

\([1_b, 1_a, 2, 3, 4, 5, 6, 8, 10, 18, 19, 23]\).

**false**

21. Quicksort achieves the minimum possible number of comparisons on the list \([4, 1, 4, 6]\). You count comparisons only between two elements of a list. **true**

22. The sublist \([8, 7, 10]\) will be considered while running Quicksort algorithm on the list \([46, 8, 29, 11, 10, 7, 12]\). **false**

23. Lists \([1,4,4,6]\) and \([4,4,1,6]\) both achieve the largest possible number of comparisons while running Quicksort algorithm. You count comparisons only between two elements of a list. **false**

24. 5 will be chosen as the pivot at some step of Quicksort algorithm while sorting the list \([17, 34, 5, 8, 1, 4, 16]\). **true**

25. 

\[
\begin{align*}
T(n) &= T(n-1) + \frac{(C+3)(n-1)}{2} \\
T(1) &= 0
\end{align*}
\]

is the recurrence relation and the initial condition in the average case of the following piece of code:

Function Blah (array a[0..n-1])
if n=1 then return a
for j from 0 to n-2 do
  if a[j] < a[j+1] then swap a[j] and a[j+1]
  Else do constant number C elementary operations
Blah(a[0..n-2]) return a

Elementary operations in this question are a comparison of two elements of the list a and a swap of two elements of the list a.

**true**

26. The solution of

\[
\begin{align*}
T(n) &= T(n-2) + 2n + 1 \\
T(0) &= 0 \\
T(1) &= 1
\end{align*}
\]

is the following function:

\[
T(n) = \begin{cases} 
\frac{n(n+3)}{2}, & \text{if } n \text{ is even,} \\
\frac{(n+1)(n+2)}{2} - 2, & \text{if } n \text{ is odd.}
\end{cases}
\]

**true**

27. Solution of

\[
\begin{align*}
T(n) &= 2T\left(\left\lfloor \frac{n}{3} \right\rfloor \right) + T\left(n - 2 \left\lfloor \frac{n}{3} \right\rfloor \right) + 4 \\
T(0) &= T(1) = T(2) = 0
\end{align*}
\]

is \(\Theta(3^k)\). **true**
28. Given the recurrence relation below:

\[
\begin{align*}
T(n) &= \frac{1}{n}(T(0) + T(1) + \ldots + T(n-1)) + 5, \\
T(0) &= 0.
\end{align*}
\]

Then the solution is in $\Theta(\log_{13}(n))$. true

29. The algorithm A with running time $T(n) = C n \log_{10}(n^7)$ takes 10 second to process input of size $10^4$. Then it will take more than 10 years to process an input of size $10^{16}$. true

30. The running time of the following piece of code is $n C([\log_7(\log_3 n)] + 1) \left\lceil \frac{n - 2}{6} \right\rceil$

\[
i \leftarrow 2
\]
\[
j \leftarrow 3
\]
while $i < n$ do
    while $j \leq n$ do
        for $k$ from 1 to $n$
            constant number $C$ of elementary operations
        $j \leftarrow j^7$
    $i \leftarrow i + 6$
    $j \leftarrow 3$

Note, you count only comparisons of two elements of the list as elementary operations. true. Any answer will be worth 1 point.

31. Algorithm A takes $n^5 \log_3(n) + 10n^2$ elementary operations and algorithm B takes $17n^6 + n^4 \log_7 n$. Then for large enough input algorithm A is faster that algorithm B. true

32. The running time of the following piece of code is $\Theta(n^2 \log_2(n))$

\[
k \leftarrow 1
\]
for $i$ from 1 to $n$
    if $i$ is a power of 5 then
        while $k \leq n$
            $k$ elementary operations
            $k \leftarrow k + 1$
false

33. $f(n) = 457 \cdot n^n + 67^n + n \log_2 n$ is $O(n!)$. false

34. $f(n) = n^2 \log_2(\log_2(\log_2 n))$ is $O(n^2 \log_2(\log_2 n))$. false

35. $f(x) = \frac{45 \cdot 3^n + n^{1000000} \log_2(\log_2 n)}{n^7 + 1}$ is $\Theta(3^n)$. false
36. **Free-response:** Let \( a = [0..n-1] \) be a list with \( n \) elements, such that \( n \) is a power of 2. You know that this list has the following property:

if you run the mergesort on this list every time you split into two sublists one of them is ordered and the other one has elements which are strictly larger than any element of the sorted sublist.

a) Write the recurrence relation and initial conditions for the mergesort algorithm on lists satisfying the property above. In this question you count as elementary operations comparisons of two elements of lists and writing in a list.

You need to consider only lists which have the property above. These property will hugely affect the merging operation.

Since \( n \) is a power of two you always split into two sublists of the same size.

Suppose you have two sublists \( l[0..k-1] \) and \( r[0..k-1] \) of size \( k \). Without loss of generality you can think that the list \( l \) is sorted and for every \( 0 \leq i \leq k-1, 0 \leq j \leq k-1 \) you have \( l[i] \leq r[j] \). The two pointers will start at the beginning of two lists. Because for every \( 0 \leq i \leq k-1, 0 \leq j \leq k-1 \) \( l[i] \leq r[j] \) every element of \( l \) with be compared to \( r[0] \) and then written to the final list. So you will do \( k \) comparisons and \( k \) writes. After that \( r \) will be added to final list, that is, you need to add \( k \) writes. In total there are \( 3k \) elementary operation to merge two lists of size \( k \).

Let \( T(n) \) be the number of elementary operations needed to mergesort a list of size \( n \) with the property above.

**Approach 1:** If you use the pseudo-code from the course book then:

\[
T(n) = n + 2T\left(\frac{n}{2}\right) + n + 3\frac{n}{2}.
\]

That is the recurrence relation and the initial condition are

\[
\begin{align*}
T(n) &= 2T\left(\frac{n}{2}\right) + 7\frac{n}{2} \\
T(1) &= 0
\end{align*}
\]

**Approach 2:** If you use the pseudo-code from the slides then:

\[
T(n) = 2T\left(\frac{n}{2}\right) + 3\frac{n}{2}.
\]
That is the recurrence relation and the initial condition are

\[
\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + \frac{3n}{2} \\
T(1) &= 0
\end{aligned}
\]

b) Solve the recurrence relation from a)

**Approach 1:** Let \( n = 2^k \) for some positive integer \( k \). Then

\[
\begin{aligned}
T(2^k) &= 2T(2^{k-1}) + 7 \cdot 2^{k-1} = 2(2T(2^{k-2}) + 7 \cdot 2^{k-2}) + 7 \cdot 2^{k-1} \\
&= 2^2T(2^{k-2}) + 7(2^{k-1} + 2^{k-1}) = 2^2(2T(2^{k-3}) + 7 \cdot 2^{k-3}) + 7(2^{k-1} + 2^{k-1}) = \\
&= 2^3T(2^{k-3}) + 7(2^{k-1} + 2^{k-1} + 2^{k-1}) = 2^3T(2^{k-3}) + 7 \cdot 3 \cdot 2^{k-1} = \ldots = \\
&= 2^iT(2^{k-i}) + 7 \cdot i \cdot 2^{k-1} = \left| i = k \right| = 2^kT(2^{k-k}) + 7 \cdot k \cdot 2^{k-1} = \\
&= 2^kT(1) + 7 \cdot k \cdot 2^{k-1} = 2^k \cdot 0 + 7 \cdot k \cdot 2^{k-1} = 7 \cdot k \cdot 2^{k-1} = \\
&= \frac{7k2^k}{2} = \frac{7n \log_2 n}{2}
\end{aligned}
\]

So the solution is \( T(n) = \frac{7n \log_2 n}{2} \).

**Approach 2:** Following the same step as in Approach 1 the solution is \( T(n) = \frac{3n \log_2 n}{2} \).

c) give the best asymptotic bound of the solution from b)

In both approaches \( T(n) \) is in \( \Theta(n \log_2 n) \). To see it you need to consider the limit:

\[
\lim_{n \to \infty} \frac{3n \log_2 n}{2} \frac{2}{n \log_2 n} = \frac{3}{2}.
\]

Since the answer is a constant between 0 and infinity the bound is correct.