Part 1: Multiple Choice

Each multiple choice question is worth **2.5 marks**. Clearly mark your answers. You can choose one or two answers for each question:

- If you select only one answer for a question: if the answer is correct you will get 2.5 marks; otherwise you will get 0 marks.
- If you select two answers: if one of the answers is correct, you will get 1.3 marks; otherwise you will get 0 marks.
- If you select more than 2 answers, you will get 0 marks.

Important note: When asked to compute the time complexity of an algorithm, you need to give the tightest characterization of the order of the time complexity. So, for example, if the time complexity of an algorithm is $f(n) = cn$ for some constant $c > 0$, while it is true that $f(n)$ is $O(n^2)$, this answer will be considered incorrect, as the tightest order of $f(n)$ is $O(n)$.

1. Algorithms `findAVL(k)` and `find24(k)` implement the `get(k)` operation from the ordered dictionary ADT using an AVL tree and a (2,4) tree, respectively. Program P_{AVL} implements `findAVL` in C++ and program $P_{(2,4)}$ implements `find24` in Java. The same collection S of n records is stored in an AVL tree given as input to P_{AVL} and in a (2,4) tree give as input to $P_{(2,4)}$. Both programs are executed to search for a data item with key $k \notin S$ in their respective trees and their running times are measured.

   The experiment then is repeated for many sets S of different sizes n, but always $k \notin S$. Which if the following statements is true?
   (A) P_{AVL} is always faster than $P_{(2,4)}$ regardless of the value of n.
   (B) $P_{(2,4)}$ is always faster than P_{AVL} regardless of the value of n.
   (C) P_{AVL} is always faster than $P_{(2,4)}$ for very large values of n.
   (D) $P_{(2,4)}$ is always faster than P_{AVL} for very large values of n.
   (E) Depending on the computer used to execute the programs, $P_{(2,4)}$ could be faster than P_{AVL} or P_{AVL} could be faster than $P_{(2,4)}$.

2. The values 9, 5, 2, 1 are inserted, in the given order, in a hash table of size 7 and hash function $h(k) = k \bmod 7$. Collisions are resolved using double hashing with secondary hash function $h'(k) = 5 - (k \bmod 5)$. In which entry of the table is the key 1 stored?

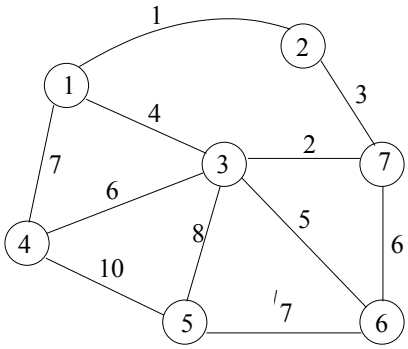
   (A) 1
   (B) 2
   (C) 3
   (D) 5
   (E) 6

3. How many different AVL trees containing the keys 1, 2, 3, 4 can be built? Each key must appear once in each AVL tree.

   (A) 2
   (B) 4
   (C) 6
   (D) 8
   (E) 14

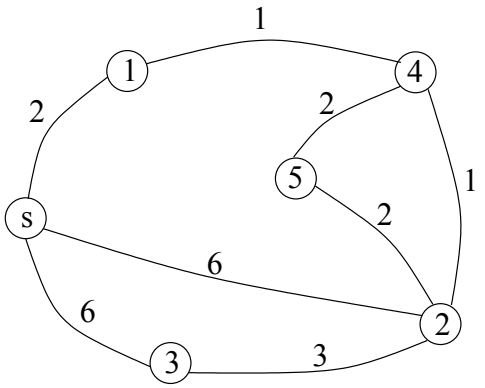
4. Consider the following graph. Which edges, and in which order, are selected by the algorithm of Prim and Jarník if it starts at vertex 1?

(A) (1,2), (2,7), (3,7), (1,3), (3,6), (3,4)
(B) (1,2), (3,7), (2,7), (3,6), (3,4), (5,6)
(C) (1,2), (2,7), (3,7), (3,6), (6,7), (3,4)
(D) (1,2), (2,7), (3,7), (3,6), (3,4), (5,6)
(E) (1,2), (3,7), (2,7), (3,6), (1,3), (5,6)



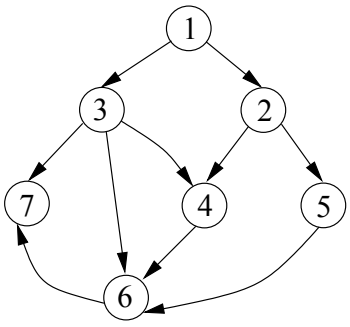
5. Consider the following weighted graph. Assume that we use Dijkstra's algorithm to find shortest paths from vertex s to the other vertices in the graph. Which edges and in which order are selected by Dijkstra's algorithm?

(A) $(s,1)$, $(1,4)$, $(2,4)$, $(4,5)$, $(s,3)$
(B) $(s,1)$, $(s,2)$, $(s,3)$, $(1,4)$, $(4,5)$
(C) $(s,1)$, $(1,4)$, $(4,5)$, $(2,4)$, $(s,3)$
(D) $(s,1)$, $(1,4)$, $(2,4)$, $(s,3)$, $(4,5)$
(E) $(s,1)$, $(1,4)$, $(2,4)$, $(4,5)$, $(2,3)$



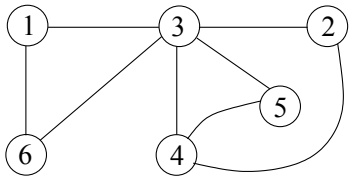
6. Which of the following statements is true for all possible ways of performing a depth first search traversal (DFS) of the following graph starting at vertex 1? Remember that in a directed graph an edge can only be traversed in the direction specified by the arrow so, for example, in the figure below we can go from 2 to 5 but not from 5 to 2.

(A) Vertex 7 cannot be the last vertex visited.
(B) Vertex 3 must be visited before vertex 6.
(C) Vertex 7 cannot be visited immediately before vertex 6.
(D) Vertex 6 cannot be visited before vertex 5.
(E) Vertex 2 must be the second vertex visited.



7. Consider the following graph. Which one of the following **is NOT** a valid order in which the vertices of the graph can be visited by a breadth first search traversal (BFS)?

(A) 1, 6, 3, 5, 4, 2
(B) 5, 4, 3, 2, 6, 1
(C) 4, 3, 2, 6, 5, 1
(D) 2, 3, 4, 6, 5, 1
(E) 3, 6, 4, 1, 2, 5



8. A database containing 10^{12} data items is to be stored in disk. Each data item uses 10 bytes and each reference to a child node uses 10 bytes as well. Two hard disks are available for storing the database. Disk A has blocks of size 100 bytes and disk B has blocks of size $10,000$ bytes. Each time that a disk is accessed a whole block is copied into memory. Which of the following data structures would allow the `get(key)` operation to be performed as efficiently as possible in the worst case? (Recall that $\log_{10^a} 10^b = b/a$ and $\log_2 100 \approx 7$.)

(A) A B-tree stored in disk A and in which every node has size $1,000$ bytes.
(B) A B-tree stored in disk B and in which every node has size $10,000$ bytes.
(C) A B-tree stored in disk A and in which every node has size $10,000$ bytes.
(D) A sorted array stored in disk B
(E) The B-trees specified in (A), (B), and (C) are equally efficient and they all are more efficient than a sorted array.

9. Let G be an **undirected, connected** graph. Let u be a vertex of G. Consider the following algorithm. At the beginning all vertices of G are unmarked.

Algorithm $B(G, u)$
    Mark u
    $c \leftarrow 0$
    for each edge (u, v) incident on u **do** {
        $c \leftarrow c + 1$
        if v is not marked **then** $c \leftarrow c + B(G, v)$
    }
    return c

What does the algorithm do?
(A) It computes the number of edges in G.
(B) It computes the sum of the degrees of all the vertices in G.
(C) It computes the number of edges in the path from u to v.
(D) It computes the number of edges incident on u.
(E) It computes the number of vertices in G times the number of edges.

10. Which is the correct definition for "$f(n)$ is **not** $O(g(n))$"?

(A) $f(n) > g(n)$ for all $n \geq 1$.
(B) There are constants $c > 0$ and $n_0 \geq 1$ such that $f(n) > cg(n)$ for all $n \geq n_0$.
(C) There are constants $c > 0$ and $n_0 \geq 1$ such that $f(n) > cg(n)$ for at least one value $n \geq n_0$.
(D) For all constants $c > 0$ and $n_0 \geq 1$, $f(n) > cg(n)$ for at least one value $n \geq n_0$.
(E) For all constants $c > 0$ and $n_0 \geq 1$, $f(n) > cg(n)$ for all $n \geq n_0$.

11. The smallest (2,4)-tree of height 1 has 1 key, and the smallest (2,4)-tree of height 2 has 3 keys. How many keys does the smallest (2,4)-tree of height 4 have? ("Smallest" here means "smallest number of keys".)

(A) 5
(B) 7
(C) 15
(D) 31
(E) 32

12. What is the solution for the following recurrence equation?

$f(1) = 0$
$f(n) = 2 + f(n-1)$, for $n > 1$

(A) $4n - 4$
(B) $2 \log n$
(C) $2n - 2$
(D) $2^n - 2$
(E) $(n-1)/2$

13. A *min priority queue* is an ADT with the following operations:

- `isEmpty()`: returns `true` if the min priority queue is empty, and
- `removeMin()`: removes the smallest element from the min priority queue.

Consider the following algorithm:

Algorithm `Get`(P, x)
Input: Min priority queue P storing n values, and value x
Output: `True` if x is in P and `false` otherwise

```
while P.isEmpty() = false do {
    t ← P.removeMin()
    if t = x then return true
}
return false
```

For which one of the following implementations of the min priority queue does algorithm `Get` have the lowest time complexity in the worst case?

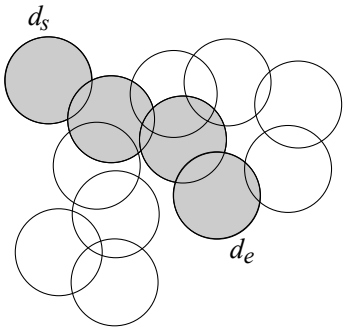
(A) An array sorted in decreasing order
(B) A $(2, 4)$-tree
(C) A multiway search tree of order d = 10
(D) A hash table with separate chaining
(E) An AVL tree

Part 2: Written Answers. Write your answers **only** in the space provided.

To answer the next two questions you can use any of the algorithms studied in class. Indicate how to model each problem with a graph, which algorithm you would use, and which changes (if any) you need to make to the algorithm to answer each question. For example, let the problem be to find a shortest way of driving from city A to city B given a road map M. Your answer might be:

- *Graph model:* Build a graph $G = (V, E)$ where every node is a city of M and every edge is a road. The length of an edge is the length of the corresponding road.
- *Algorithm:* Use Dijkstra's algorithm to compute a shortest path from node A to node B. This is the shortest way of driving from city A to city B.

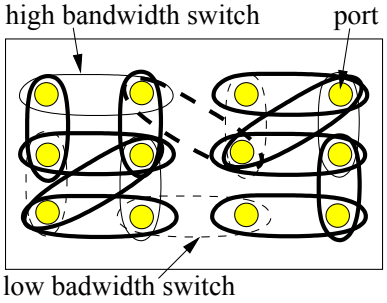
14. (5 marks) A set of coins $d_1, d_2, \ldots d_n$ are placed on a flat surface so that some of them overlap. Describe an algorithm to compute a path from coin d_s to coin d_e such that any two coins along the path overlap. This path must have the minimum possible number of coins in it. For example, for the following set of coins the desired path is shaded.



(2 marks) *Graph model:* Build a graph G=(V,E) where each coin is a node and there is an edge between two vertices if their corresponding coins overlap.

(3 marks) *Algorithm:* Use breadth first search to find a path with minimum number of edges from the node corresponding to d_s to the node corresponding to d_e. We can also use Dijkstra's algorithm if every edge of G is assigned length 1.

15. (5 marks) A circuit board consists of a set P of ports and a set S of switches. Each switch $s_i = (p_i, p_j)$ connects two ports, p_i and p_j. A port p_i can send a message to a port p_j if there is a set of switches $(p_i, p_{a1}), (p_{a1}, p_{a2}), \ldots (p_{ak}, p_j)$ providing a communication path between p_i and p_j. Assume that the set S of switches creates at least one path between every pair of ports. Let there be two types of switches *low bandwidth* and *high bandwidth*. Describe an algorithm that finds the smallest set $T \subseteq S$ of switches that allows every port to send messages to every other port, so that the total number of low bandwidth switches is minimum. For example, for the following circuit board the solution is given by the set of switches in bold.



(2 marks) *Graph model:* Build a graph G=(V,E) where each port is a node and each switch is an edge. An edge representing a high bandwidth edge has length 1 and an edge representing a low bandwidth edge has length n, where n is the number of ports.

(3 marks) *Algorithm:* Use the algorithm of Prim and Jarník to compute a minimum spanning tree of G.

16. The following questions refer to the time complexity of the following algorithm.

Algorithm `traverse`(G, u)
Input: Undirected, connected graph G, and vertex u of G.

Mark u } c_3
For each edge (u, v) incident on u **do** {
    `process`(u, v)
    if v is not marked **then** `traverse`(G, v) } c_2
} } $(c_1 n + c_2) \times \text{degree}(u)$

Algorithm `process`(u, v) performs $c_1 n$ operations, where c_1 is a constant value. Assume that G is stored in an adjacency list:

i) (1 mark) Ignoring recursive calls, how many operations are performed every time that the algorithm is invoked? Explain (as explanation you may annotate the different parts of the algorithm witht the number of operations that they perform).

Each iteration of the for loop performs $c_1 n + c_2$ operations. The for loop is repeated degree(u) times and outside the loop a constant number c_3 of operations are performed.
Hence, in each call the algorithm performs

$$c_3 + (c_1 n + c_2) \times \text{degree}(u)$$

operations.

ii) (2 marks) How many calls does the algorithm perform? Explain.

Since the graph is connected, the algorithm calls itself recursively once per unmarked vertex. Therefore, one call is made per node so the total number of calls is n.

iii) (2 marks) What is the total number of operations performed by the algorithm? Explain.

$$\sum_{u \in G} (c_3 + (c_1 n + c_2) \times \text{degree}(u)) =$$

$$c_3 n + (c_1 n + c_2) \sum_{u \in G} \text{degree}(u) = c_3 n + (c_1 n + c_2)(2m)$$

$$= c_3 n + 2 c_1 n m + 2 c_2 m$$

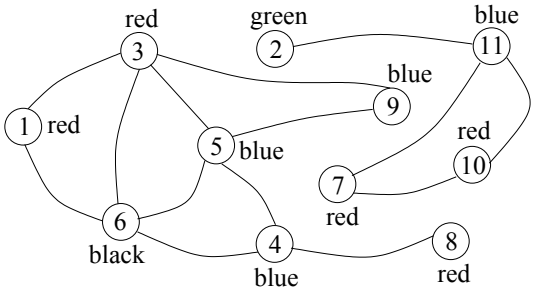
iv) (0.5 marks) What is the order of the time complexity? Explain.

Ignoring constants and since $nm > n$ and $nm > m$, the time complexity is $O(mn)$

17. (13 marks) Let $G = (V, E)$ be an undirected graph in which every node is assigned a color. Write in pseudo-code a **recursive algorithm** that given a node s it returns the number of edges (u, v) that can be reached from s such that the color of u is different from the color of v. Use u.color to denote the color of vertex u.

For example, for the following graph and vertex $s = 1$, the algorithm must return the value 7 because edges $(1, 6), (3, 6), (3, 5), (4, 6), (5, 6), (3, 9)$, and $(4, 8)$ are reachable from s and their endpoints have different colors; if $s = 2$ the algorithm must return the value 3.

Hint. Perform a DFT traversal of G starting at s, but make it sure you do not count edges more than once.



Algorithm count_edges(s)
In: Vertex s of an undirected graph $G = (V, E)$ whose vertices are colored.
Out: Number of edges reachable from s whose two endpoints have different colors.
{

```
Mark(s)
c ← 0
For each edge (s, u) do
    if u is not marked then {
        if s.color ≠ u.color then c ← c+1
        c ← c + count_edges(u)
    }
return c
```

}

18. (12 marks) Let T and R be two AVL trees, each storing n integer keys. All keys are different. Write in pseudocode an algorithm that stores in an array A of size $2n$ all the keys of T and R in increasing order of value.

THE ONLY OPERATIONS that you can use on the AVL trees are:

- `smallest()` that returns the key with smallest value in the AVL tree
- `remove(key)` that removes the specified key from the tree, and
- `isEmpty()` that returns true if the AVL tree has no data items, and false otherwise.

If you use additional operations to manipulate the AVL trees you will not get full marks.

Algorithm AVLsort(T, R, A, n)
Input: AVL trees T and R each storing n records; empty array A of size $2n$.

```
{
C3  { i ← 0
      K1 ← T.smallest()
      K2 ← R.smallest()
while (T.isEmpty() = false) and (R.isEmpty() = false) do {
    if K1 < K2 then {
        A[i] ← K1
        T.remove(K1)
        if T.isEmpty() = false then K1 ← T.smallest()
    } else {
        A[i] ← K2
        R.remove(K2)
        if R.isEmpty() = false then K2 ← R.smallest()
    }
    i ← i+1
}
if T.isEmpty() then
    while R.isEmpty() = false do {
        K2 ← R.smallest()
        A[i] ← K2
        R.remove(K2)
        i ← i+1
    }
else while T.isEmpty() = false do {
        K1 ← T.smallest()
        A[i] ← K1
        T.remove(K1)
        i ← i+1
    }
return A
}
```

n, C1 log n
C1 log n
(n-n1) C2 log n
C2 log n
(n-n1) C2 log n
C2 log n

19. (3.5 marks) Compute the time complexity of your algorithm for the previous question in the worst case. **Explain** how you computed the time complexity and give the order of the time complexity.

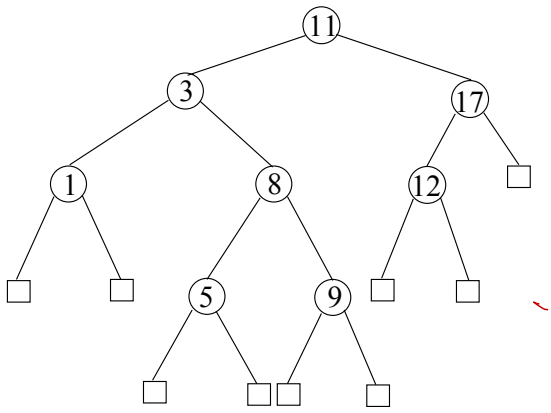
The time complexity of remove() and smallest is $O(\log n)$. Therefore, each iteration of the first while loop performs $c_1 \log n$ operations. Each iteration of the other two loops performs $c_2 \log n$ operations.

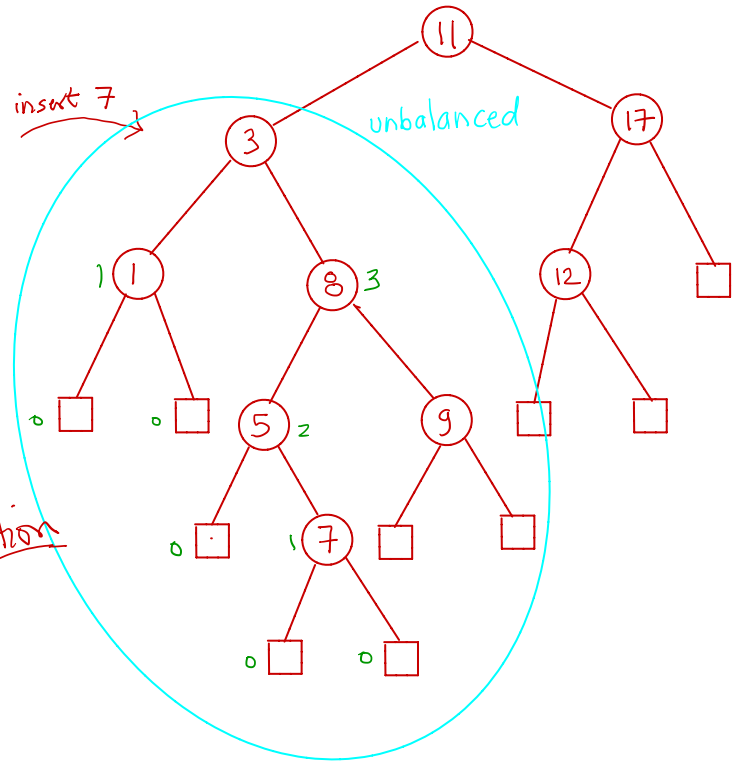
The first loop performs n_1 iterations and either the second or the third while loops perform $n - n_1$ iterations. Outside the loops a constant number c_3 of operations are performed. Therefore, the total number of operations performed by the algorithm is

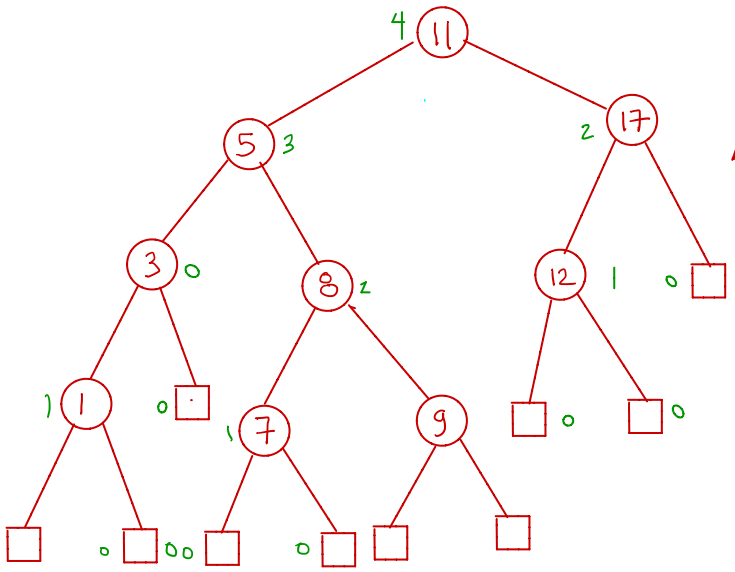
$$c_3 + c_1 n_1 \log n + c_2 (n - n_1) \log n$$

Discarding constants we get that the time complexity is $O(n_1 \log n + (n - 1) \log n)$ which is $O(n \log n)$.

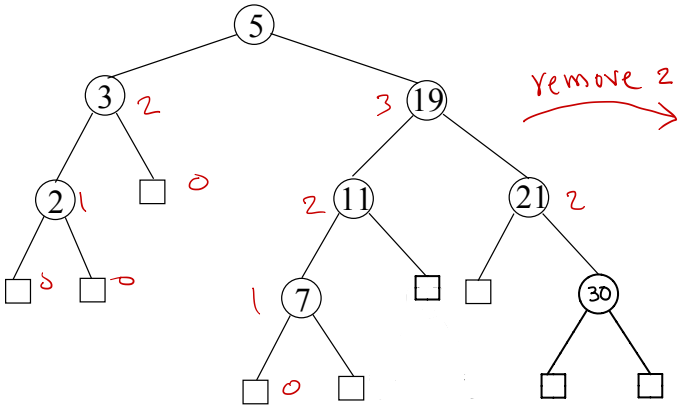
20. (5 marks) Consider the following AVL tree. Insert the key 7 into this tree and re-balance if needed. You **must** use the algorithms discussed in class for inserting data into an AVL tree. Show all intermediate trees.

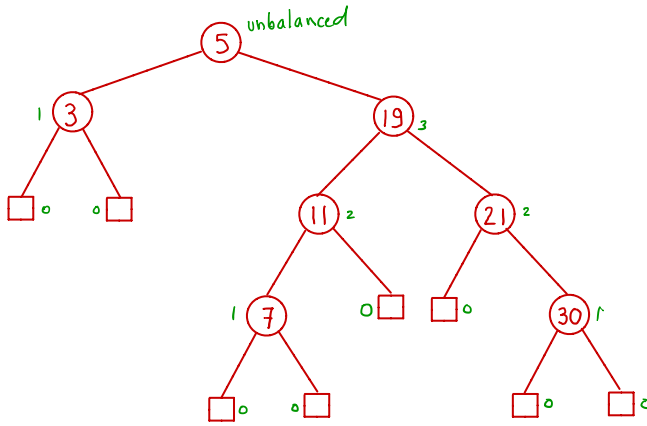


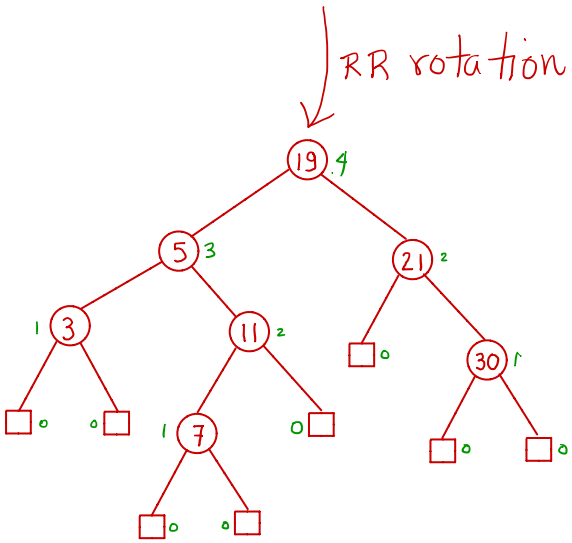




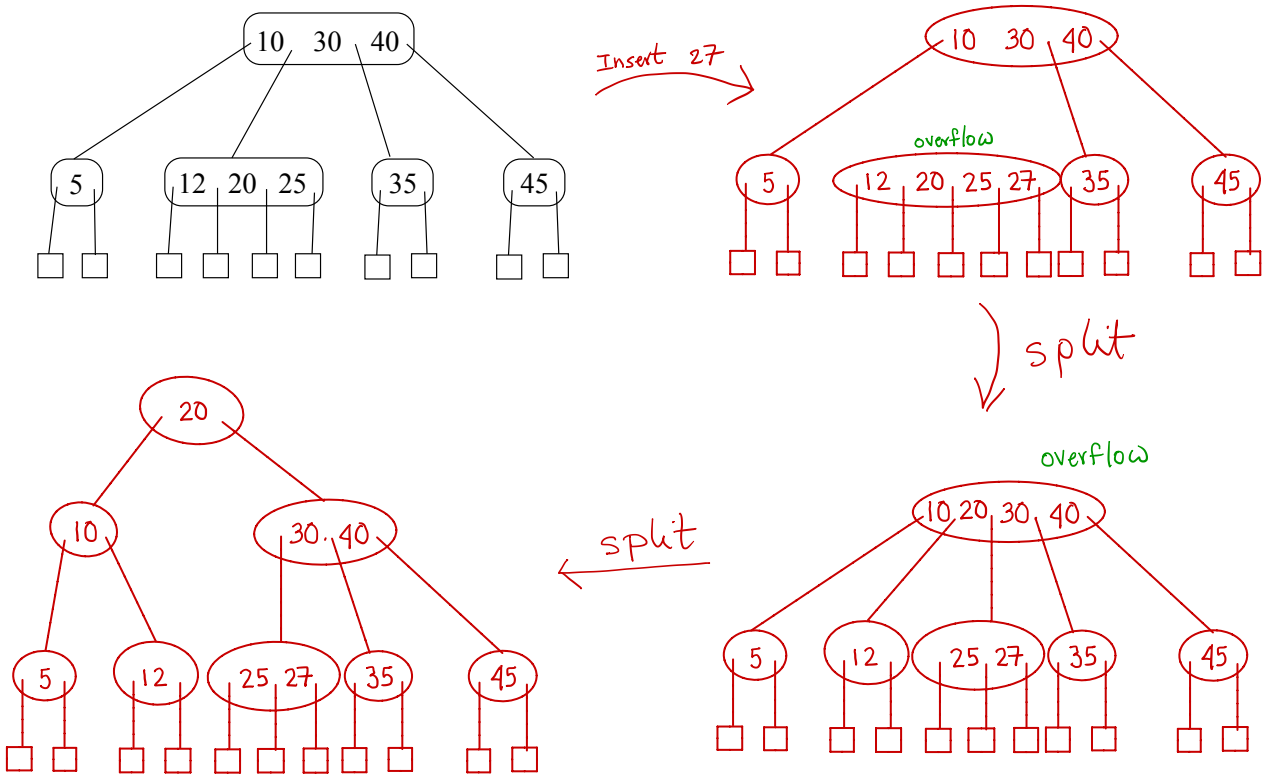
21. (5 marks) Consider the following AVL tree. Delete the key 2 and re-balance the tree if needed. You **must** use the algorithms discussed in class for removing data from an AVL tree. Show **all** intermediate trees.







22. (5 marks) Insert the value 27 in the following $(2, 4)$ tree. You **must** use the algorithm discussed in class for inserting information in a $(2, 4)$ tree. Show **all** intermediate trees. Indicate whenever a split, fusion, or transfer operation is required.



23. (6 marks) Delete the value 45 from the following $(2,4)$-tree. You **must** use the algorithm discussed in class for removing information from a $(2,4)$ tree. Show **all** intermediate trees **including the tree right after deleting 45**. Indicate whenever a split, transfer or a fusion operations are required.