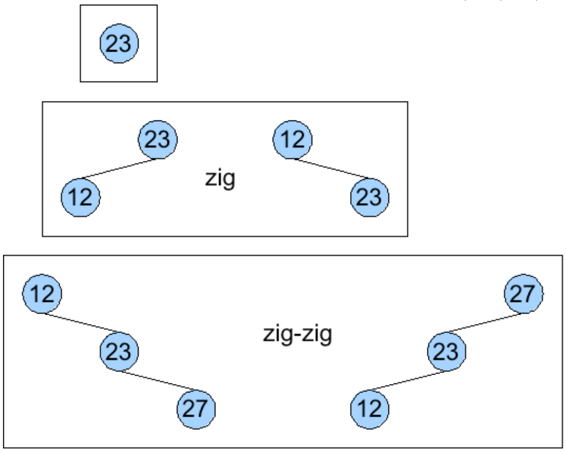
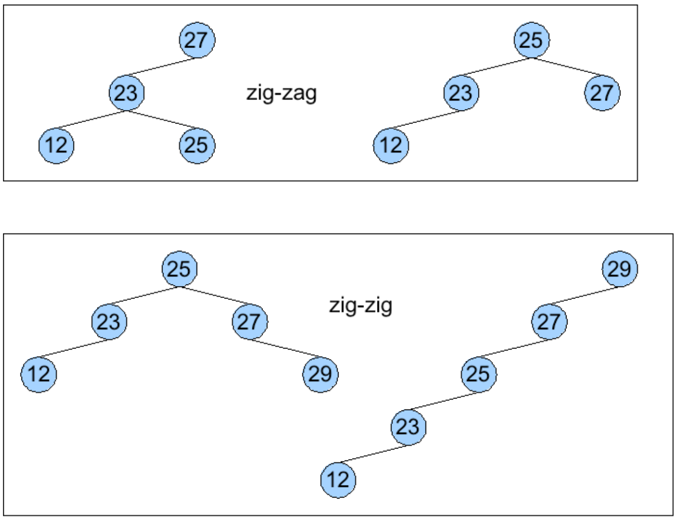
Question 1
Show the structure of an initially empty splay tree after inserting each of the keys in the following sequence:

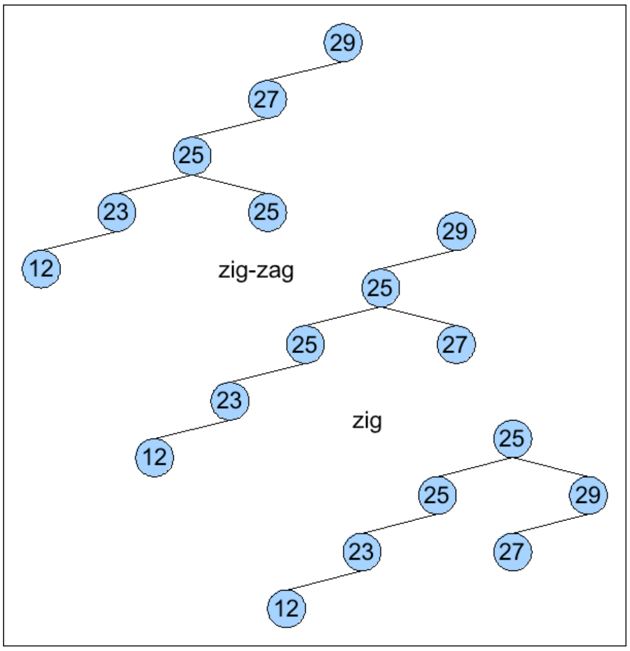
23, 12, 27, 25, 29, 25



Question 1
Show the structure of an initially empty splay tree after inserting each of the keys in the following sequence:

23, 12, 27, 25, 29, 25





Question 2
Briefly describe why accessing nodes of a splay tree have amortised time-complexity of O(log *n*). Refer to section 10.3.3 of the text book.

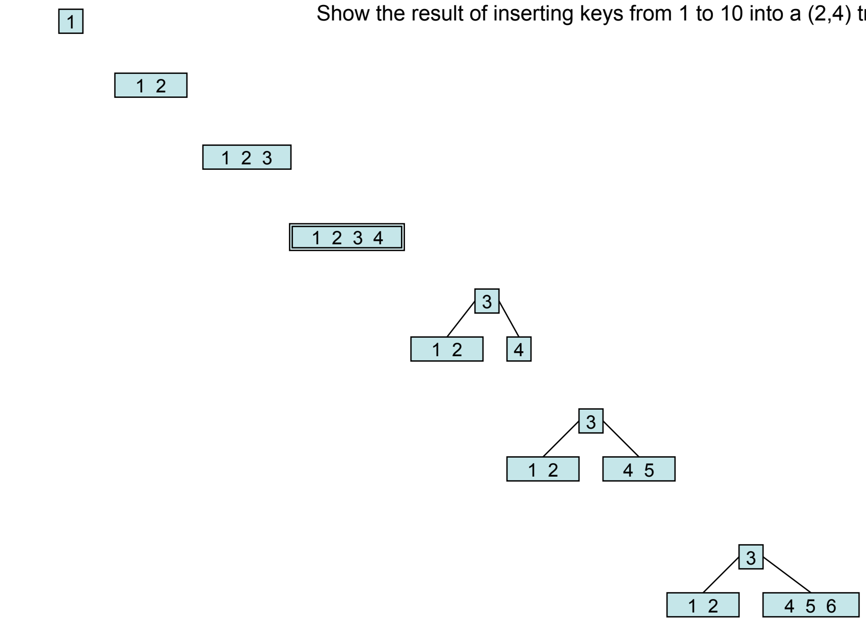
- The proof considers the cost of each splay – each zig, zig-zag, or zig-zig of a splay is referred to as a *substep and cost 1, 2 and 2 cyber dollars respectively*.
  - Let *T* be a splay tree with *n* keys
  - $n(v)$ is the number of keys in the subtree rooted at v
  - Rank – $r(v) = \log(n(v))$

- The proof considers payment of cyber dollars for each splay. Each node of the tree has an account associated with it that can store cyber dollars. **The tree has an invariant that before and after splaying, each node *v* of T has r(*v*) cyber dollars**. When a splay is performed costing, *X,* an amount *Y* is paid, the difference *Z* is then either paid to, or removed from the splay tree.

- The proof shows that only an amount O(log *n)* is required to pay for each splay and maintain the invariant.

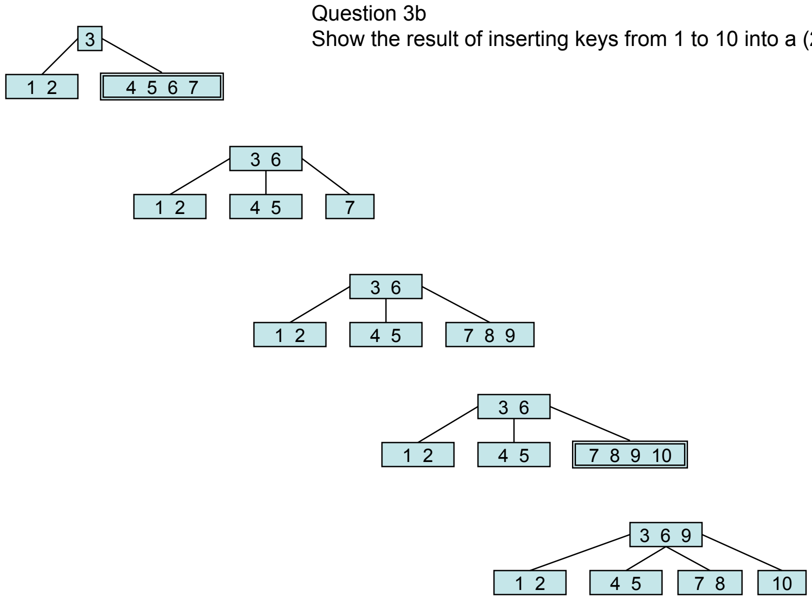
Question 3a

Show the result of inserting keys from 1 to 10 into a (2,4) tree



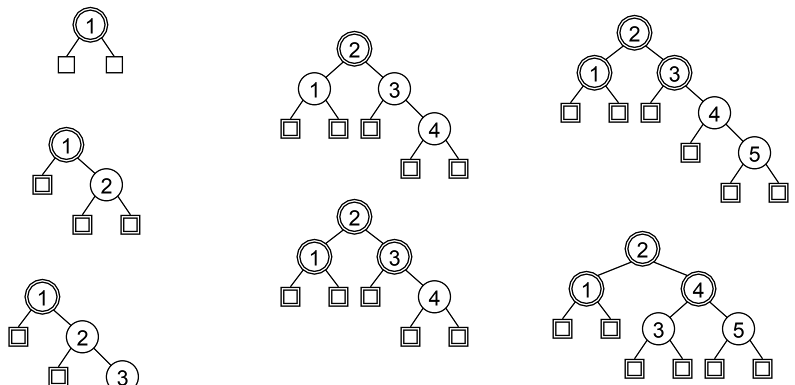
Question 3b

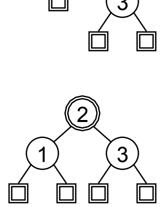
Show the result of inserting keys from 1 to 10 into a (2,4) tree



Question 3b

Show the result of inserting sequential keys from 1 to 10 into a red-black tree.



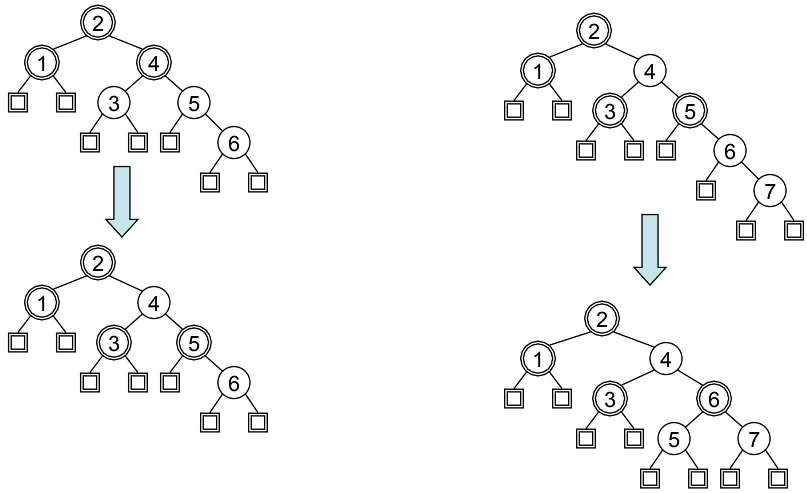


Root property: the root is black
External property: every leaf is black
Internal property: the children of a red node are black
Depth property: all the leaves have the same black depth

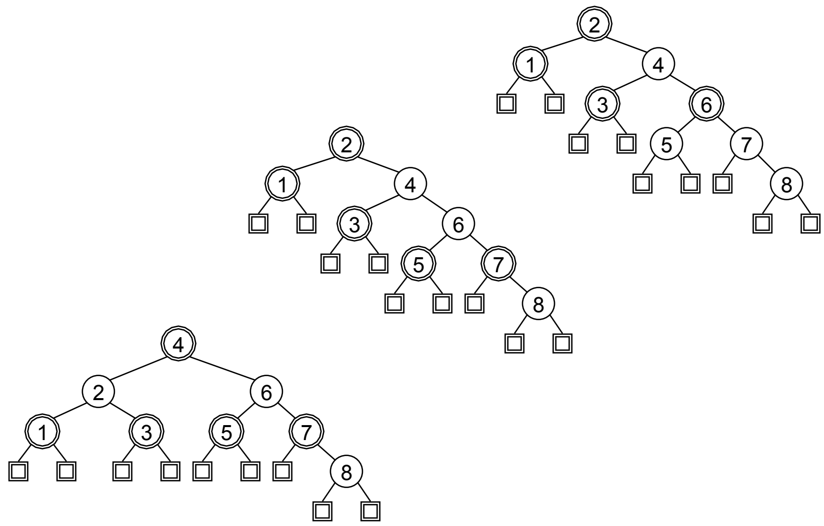
Insertion: insert as with a binary search tree with node colored red. *Double red violation* either handled with **restructure** if parent node's sibling is black; or **recolor** if parent node's sibling is red.

Question 3b

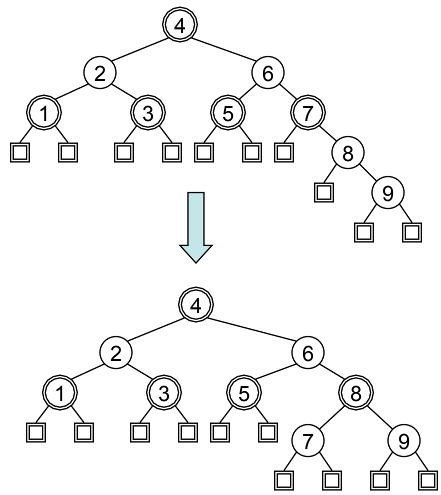
Show the result of inserting sequential keys from 1 to 10 into a red-black tree.



Question 3b
Show the result of inserting sequential keys from 1 to 10 into a red-black tree.

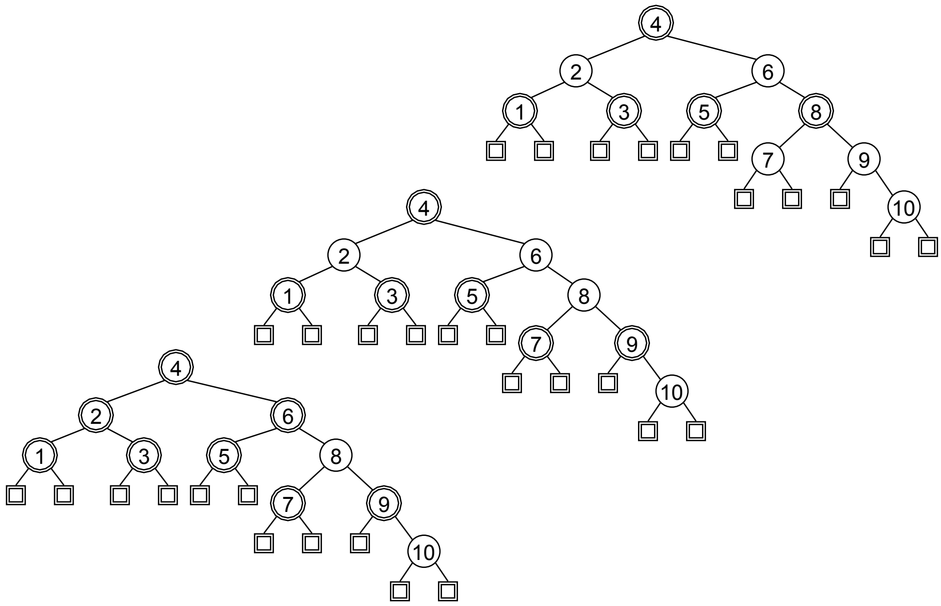


Question 3b
Show the result of inserting sequential keys from 1 to 10 into a red-black tree.



Question 3b

Show the result of inserting sequential keys from 1 to 10 into a red-black tree.

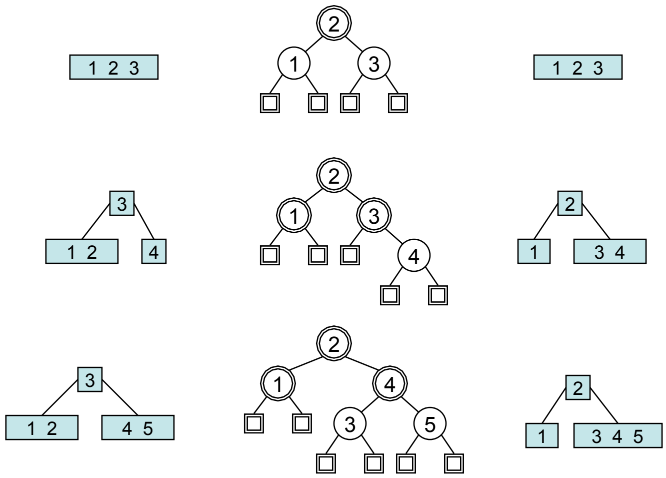


Question 4

Compare the (2,4) tree to the red-black tree.

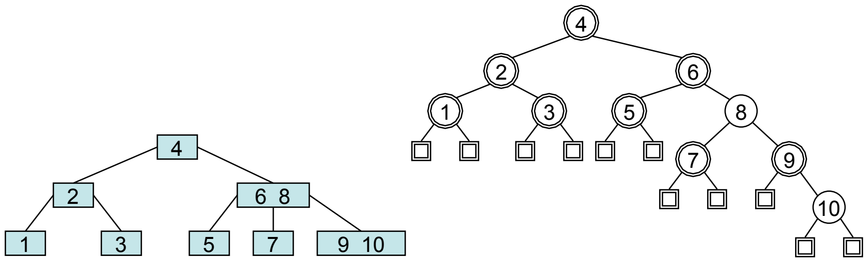
i) Can you show which nodes in the (2,4) tree correspond to the nodes in the red-black tree.

ii) Do the two methods directly correspond?



Question 4
Compare the (2,4) tree to the red-black tree.

i) Can you show which nodes in the (2,4) tree correspond to the nodes in the red-black tree.
ii) Do the two methods directly correspond?



This diagram shows that even though identical insertions into (2,4) and red-black trees don't yield directly corresponding trees, a valid red-black tree can be mapped directly to a valid (2,4) tree.

Each black node corresponds to a node in the (2,4) tree. Each red child of a black node is a member of that corresponding node.