1) Write a method that receives the root of a binary tree and returns a node that has at least one
child but no grandchildren. If no such node exists in the tree, return null.
2) Write a method that receives the root of a binary tree and returns a boolean value as to whether
the binary tree is a binary search tree.
3) Assuming you implemented hashing with chaining. Although findmax is not an operation we
usually expect in such a data structure, write a method that performs findmax. What is the time
complexity of this operation?
4) Build a heap using the sequence of insertions 4, 8, 3, 9, 5, 7, 1, 2, 6, 10, 12, 11, 4, 8, 3, 9, 5.
5) Build a heap using the make-heap operation 4, 8, 3, 9, 5, 7, 1, 2, 6, 10, 12, 11, 4, 8, 3, 9, 5.
6) Write a method that receives a heap implemented as an array and prints the heap with inorder
traversal.
7) Write a method that receives an integer n and builds and returns an array S of length n
representing two disjoint sets. (We are using the tree representation of a set where the root is
the representative, and where S[i] is the parent of i. When i is the root, S[i] is -1.) One of the
sets must contain all even numbers (0,2,...) and the other must contain all odd numbers (1,3,...)
8) Perform the following operations on a disjoint set data structures for 0..9 using trees with
union-by-rank and path compression. When the trees have the same rank, make the second tree
a child of the first tree root. Union(1,3), union(4,6), union(1,9), find(9), union(2,8), union(5,7),
union(2,7), find(7), union(1,2), find(8).