- Don't forget to set your Eclipse workspace and working set.
- You must submit the JAR file, exported (with source code), from your Eclipse project.
- You must check your JAR file to make sure all the source files (.java files) are present. It can be opened with file compression programs such as 7-zip or Winrar.
- Failure to export properly will result in your work not getting marked.

To submit:

Export your project to a JAR file, with source code.

Name your JAR file ID_Week16_Q2.jar. For example, 6623110021_Week16_Q2.jar

Submit the JAR file on MyCourseville.

(11 marks) A Treap is a type of binary search tree. Each of its node is defined as follows:

```
public class TreapNode {
    int bstValue; // value stored in the node.
    int heapValue; //priority value as in a min heap

    TreapNode left; //pointer to lower left node.
    TreapNode right; //pointer to lower right node.
    TreapNode parent; //pointer to the node above.
```

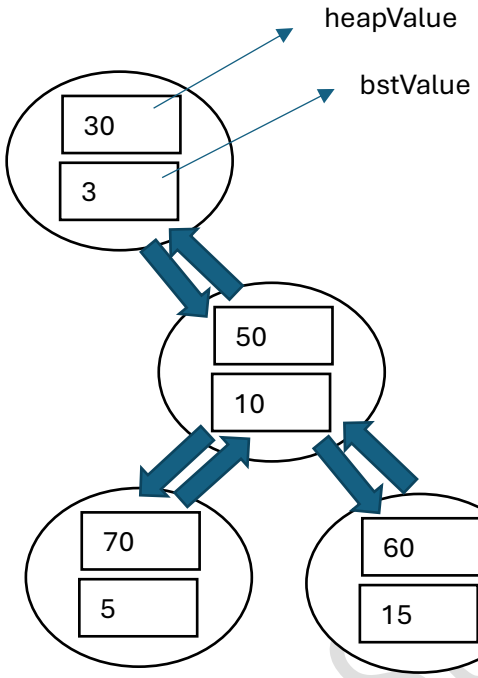
Assume that

- All (bstValue, heapValue) pair are unique.
- There can only be one heapValue for one bstValue.
- Treap does not store duplicated (bstValue,heapValue) pair.

Treap has the following properties:

- A Treap must be a binary search tree according to the bstValue in every node.
- From top (root) to bottom, the node is arranged according to heapValue (smaller value is nearer to the root).
- Treap is not a complete binary tree.

An example Treap looks like:



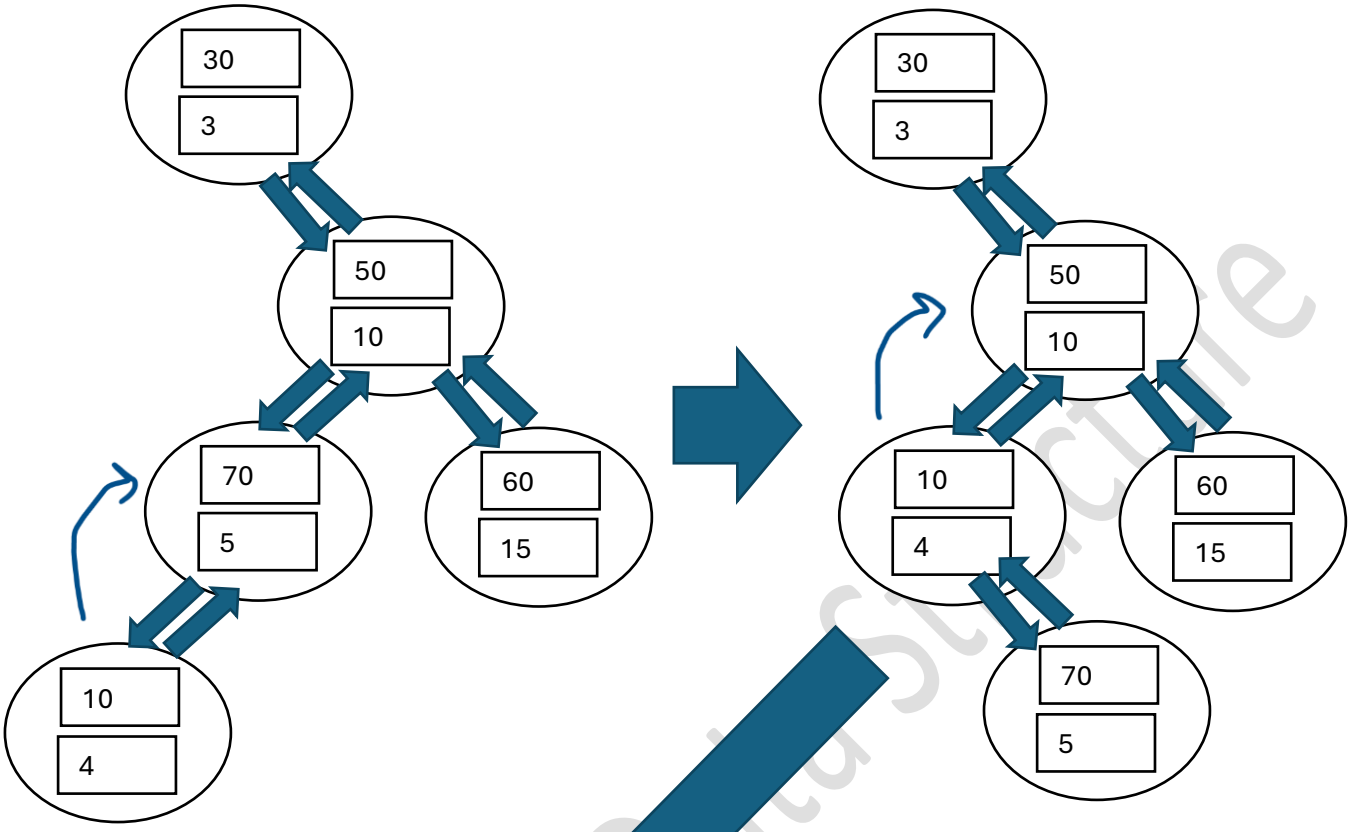
You are given codes for Binary Search Tree and AVL Tree (just 1 class for AVL tree), TreapNode, and a test file, TestTreap (scores for test are shown in code's comment).

Write class Treap (Copy and Modify codes from given classes as necessary). For methods, you only need to write these methods:

- Constructor (no parameter).
- `public TreapNode insert(int v, int h) //v is bstValue, h is healValue`
  - if v,h is stored inside the tree, do nothing and return **null** from the method.
  - if v,h is not stored inside the tree, insert a new node that contains these values.
  - Use heapValue of the newly added node to **rotate it up the tree**, such that heapValues from root to any leaf are sorted from small to large.
  - Then return the node that we added. (It may no longer be a leaf node)

(see example next page)

For example, adding node with (bstValue,heapValue) = (4,10) to the above tree will get us:



Code Hint: Make sure that you link node above a rotated node correctly.