108-1 PDSA Final (January 6, 2020)

學號：_____ 姓名：_____

1. (12%) Please analyze the time complexity of **heapsort**.
2. (15%) Describe the **Kosaraju-Sharir algorithm** that computes the strong components of a digraph. Prove its execution time is proportional to $E + V$, where E is number of the edges and V is number of the vertices in the digraph.
3. (15%) Describe how to construct a 2d-tree when given N 2-dimensional points. What are the **worst-case** and **typical-case** complexities of using a 2d-tree in a nearest neighbor search (find the closest point among N 2-dimensional points to the query point)?
4. (16%) Please compare the **worst-case** (after N inserts) complexities of the *insert* and *search* operations for the following ST (symbol table) implementations:
   A. sequential search (unordered list)
   B. binary search (ordered array)
   C. BST (binary search tree)
   D. red-black BST
5. (15%) Describe the **sweep line algorithm** for finding all the intersections among a set of N rectangles. Analyze its time complexity.
6. (15%) Single-linkage clustering is one of several methods of hierarchical clustering. It is based on grouping clusters in bottom-up fashion, at each step combining two clusters that contain the closest pair of elements not yet belonging to the same cluster as each other. Write a program that constructs a **single-linkage clustering tree** (a binary tree) using ***Kruskal's* algorithm**. Your program should take an undirected weighted graph G as the input, and return the root of the binary tree as the output.
7. (12%) Complete the following code that implements the *left rotation* operation in LLRB (left-leaning red-black) BSTs:

   Orient a (temporarily) right-leaning red link to lean left.

```
// make a right-leaning link lean to the left
private Node rotateLeft(Node h) {
    // assert (h != null) && isRed(h.right);
    Node x = h.right;

     /* provide your code here */

    return x;
}

private class Node {
    private Key key;          // key
    private Value val;        // associated data
    private Node left, right;  // links to left and right subtrees
    private boolean color;     // color of parent link
    private int N;           // subtree count

    public Node(Key key, Value val, boolean color, int N) {
        this.key = key;
        this.val = val;
        this.color = color;
        this.N = N;
    }
}
```