1. **Fill in the Blank** (5000pts)
   Note: Items 1-4 deal with Red-Black properties.

   1. Every node is either ________________ or ________________
   2. Color-wise, the root and leaves are ________________
   3. If a node is red, then its parent is ________________
   4. All simple paths from any node x to a descendant leaf have the same number of ________________
   5. The ________________ operation is used to remove an element from a stack
   6. The element removed from a *queue* is the ________________ element added to the queue
   7. Dijkstra's algorithm produces a ________________
   8. If the next empty location is found in a squared number sequence, then you are using ________________

2. **Matching** (30,000 points)

a.	Comments	Accessed by a parameter to main.
b.	Block	In Java, integer, floating-point, Boolean, and character.
c.	Primitive Types	A sequence of statements within braces.
d.	Command-line argument	Make code easier for humans to read but have no semantic meaning.
e.	Null reference	The value of an object reference that does not refer to any object.
f.	Mergesort	A recursive nstance that can be solved without recursion.
g.	Base Case	A divide-and-conquer algorithm that obtains an O(N log N) sort.
f.	Bellman–Ford algorithm	An alg that is used to solve the positive-weighted, shortest-path problem.
f.	Dijkstra's algorithm	An alg that is used to solve the negative-weighted, shortest-path problem.

3. (A Single Point) Fill in the runtimes of each sorting algorithm.

Sorting Algo	Big Oh	Theta
Insertion Sort		
Selection Sort		
Bubble Sort		
Merge Sort		
Heap Sort		
Quick Sort		

4. (1600 pts) Give the expected Big-O runtime for the following operations on using the given data structures.

   Red-Black Tree: search, min, max, successor, and predecessor

   Bubble Sort: Best, worst, and average case

   Selection Sort: Best, worst, and average case

   Heapsort: Best, worst, and average case

5. (12 pts) **Red-Black** Below is the adjacency matrix for a Red-Black tree. The outer edges indicate the value at a given node (the first value is the root, the next two are the node's children (left to right), and so on in an inorder traversal).
   Draw the red-black tree below, indicating black nodes as boxes and red nodes as circles.

$$\left[\begin{array}{ccccccccc} & 7 & 3 & 18 & 10 & 22 & 8 & 11 & 26 \\ 7 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 18 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 10 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 22 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 8 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 26 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

   Now, insert a node x=15 into the tree. *Note: you do not need to draw out the individual step(s) used in the insertion. Instead, explain clearly what step(s) you take during the insertion, including your recoloring(s) or rotation(s) if applicable.*

6. **Trees** (5! points)Perform a preorder traversal on this tree

```
            a
          /   \
        c       g
      /   \   /   \
    x      m p     q
   /          \
  y            w
```

7. **AVL** (4! points) Perform the AVL instructions listed below. Perform correct rotations.

a. Remove 10

```
           10
          /   \
        7       15
      /   \    /   \
    3      8  12    28
   /
  2
```

b. Insert 58

```
           50
          /   \
        32      70
      /   \    /
    5      37 63
```

8. **AVL** (6? points) Do that again. Perform correct rotations.

b. Insert 1

```
           74
          /   \
        32      87
      /   \    /   \
    14     59 77    90
   /
  2
```

9. (10 or 18 points) **Heaps**

   (a) Describe the structure of a min-heap, how do each nodes compare to their children?

   (b) Fill the runtimes of min-heap operations

GetMin()	________
RemoveMin()	________
Insert()	________

   (c) Fill in the array index locations of each node

The Root	________________
Node i's Left Child	________________
Node i's Right Child	________________
Node i's Parent	________________

10. **Hash Slinging Slasher** (π points) Insert the following into a hash table using linear probing with a hash function H(x) = k mod 7. $[76, 93, 40, 47, 10, 55]$

    What is quadratic probing?

11. **Code** (2^e points) Describe the runtime of the following psedocode functions.

```
def ILoveOwls(n):
      result = 0
      while n > 1:
            n //= 2
            result += 1
      return result
```

```
def IReallyLoveOwls(n)
      result = 0
      for i in range (n):
            for j in range (i, n)
                  for k in range (n):
                        result += 1
      return result
```