Problem 1

Consider the following (partial) implementation of a binary search tree and the BUILD-BST algorithm:

```
BST {
    int data
    BST left, right

    BST(int d)
        data = d
        left = null
        right = null

    void add(int toAdd)
        if toAdd < data
            if left is empty
                left = BST(toAdd)
            else
                left.add(toAdd)
        else
            if right is empty
                right = BST(toAdd)
            else
                right.add(toAdd)
}

BUILD-BST(list)
    r = BST(list[0])
    for i=1 to list.length - 1 do
        r.add(list[i])
    return r
```

Assume that *list* is a nonempty list of n elements.

a) Prove that the best case running time of BUILD-BST is $\Omega(n \log(n))$

b) Prove that the worst case running time of BUILD-BST is $\mathcal{O}(n^2)$

c) How would an AVL-tree improve the above algorithm? (ie balance, running-time of BUILD-BST, etc) Explain.

Problem 2

Give a $\mathcal{O}(n \log k)$-time algorithm that merges k sorted lists with a total of n elements into one sorted list. (Hint: use a heap to speed up the trivial $\mathcal{O}(kn)$-time algorithm)

Problem 3

Give a $\mathcal{O}(n)$ average case running time algorithm that returns the value of the kth smallest element in a list of length n.

Problem 4

Consider the following recurrence relation:

$$T(n) = \begin{cases} 1 & n = 1 \\ 7T(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

a) Use the master theorem to show that $T(n) \in \Theta(n^{\log_2(7)})$

b) Use induction to show that $T(n) = \frac{1}{6}(7n^{\log_2(7)} - 1)$

Problem 5

Let H be an empty hashtable with 9 bins that resolves collisions using chaining. Draw the state of H after the following sequence of integers have been inserted:

$$0, 39, 7, 12, 42, 4, 17, 13, 3$$

Please assume that the hashing function for an integer n is just $f(n) = n$.

Problem 6

We often refer to the number of elements in an array-list to be its *logical size* and the actual size of the array to be its current *capacity.*

Define a *contracting* array-list, to be a array-list with the constraint that the capacity of the array-list never exceeds twice its logical size.

Give an algorithm for the *add* and *remove* operations of a contracting array-list such that they are always amortized constant time.

Bonus Problem: Effecient Polynomial Multiplication

a) Give an algorithm that multiplies two degree-1 polynomials with only three multiply operations. That is, given coefficients a, b, c, d, the algorithm should compute the values of the coefficients in the expanded form of $(ax + b) \cdot (cx + d)$. (Hint: use $(a + b) \cdot (c + d)$ as one multiplication)

b) Give a divide-and-conquer algorithm for multiplying two polynomials of degree n, and prove using the master theorem that your algorithm runs in $\Theta(n^{\log_2(3)}$ time. You may assume that $n + 1$ is a power of 2.