**20CS2028 Data Structures**

**Programming assignment #4**

**Assigned on Friday July 25, 2014**

**Due on Tuesday, August 2nd, 2014 on Blackboard at 11:59PM**

**Total number of points: 35**

**Problem 1**(An Interesting Puzzle)This problem looks and should be very easy…. Assume that you are given two lists A, B of the same size, n, of positive and negative integer values. The problem is to find all the pairs of values, (a, b), a from A, b from B, such that a + b = 0.

1. **(5 points)** Describe an algorithm of complexity O(n) which solves this problem. Note that it is easy to write a O(n2) algorithm.

* Sort both lists
* Start at A[0] and B[n-1]
* Add values,
  + if negative increment A[0]-> A[1] and so on
  + if positive increment B[n-1] -> B[n-2] and so on
  + if zero you found a match increment both A and B

1. **(5 points)** Next consider that you have $k = 3, 4, …, n$, such lists. Generalize your algorithm and prove its behavior.

You would need to use a hash table and merge all the lists into one large sorted list L with the values key being equal to the list number it came from.

You then sort this list L

Next you add the first and last values in list so L[0] and L[n]

If negative increment the front side so L[0] -> L[1]

If positive increment the end side so L[n] -> L[n-1]

If zero and keys are not equal, found match, increment both front and end.

If zero and keys are equal, increment both front and end.

**Problem 2.** Consider the set of keys K = {1, 2, 3,…., 13}

1. **(5 points)** Draw a 2-3-4 tree that stores K using the fewest number of nodes.

4,8,11

12,13

5,6,7

1,2,3

9,10

1. **(5 points)** Draw a 2-3-4 tree that stores K using the maximum number of nodes.

4,8

2

10

1

3

11,12,13

6

9

5

7

1. **(5 points)** For the tree in part (a) draw four different Red-Black tree that correspond to it.

8

12

4

13

10

2

6

11

9

7

3

1

4

4

10

1

5

4

8

11

4

10

12

2

6

9

1

3

1

13

7

4

4

10

5

4

8

10

4

12

9

2

6

11

13

7

3

1

4

4

10

1

5

4

8

11

4

13

9

2

6

10

7

3

1

4

4

10

1

12

5

4

**Problem 3.** **(5 points)** Draw an example of a red-Black tree which is not an AVL tree.

8

17

4

2

6

5

1

**Problem 4.** **(5 points)** Show that any n-node binary tree can be converted into any other n-node binary tree by a sequence of O(n) rotations.

Any binary tree can be made into a left chain through rotations, so if you can make a left chain, from the left chain you can make any other binary tree.