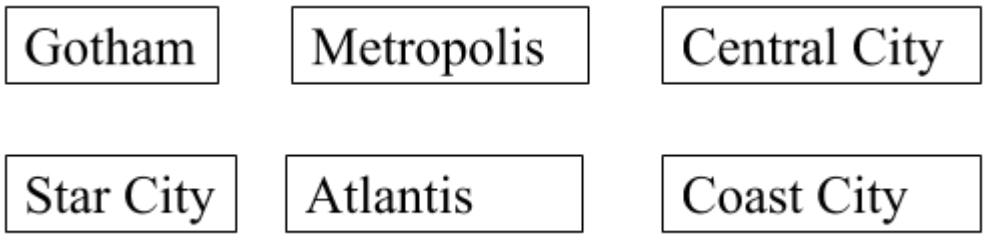
Practice Midterm 2

You have 2 hours to complete this exam. This exam is meant solely for practice and topics that are not in this exam may be covered while topics in it may not be covered. This exam is out of 100 points. Every line may have at most one statement (including closing brackets).

Problem	Points
1	12
2	5
3	6
4	7
5	8
6	12
Total	50

1 Flash Union

(a) Beary Allen is a fast UC Berkeley student wants to connect the world. He has a set of cities and wants to connect cities and see if they are connected in the fastest way possible. To do this, he will us the fastest Quick Union Data Structure that we have learned. Draw the Quick Union Data Structure in box that corresponds to how the Data Structure looks after all the calls in that box. Note that calls occurs sequentially. This means the Quick Union Data Structure in Box 2 is the starting Structure for Box 3. If there is a tie, break it by making the left tree corresponding to the left item the root.(5 pts)



connect(Gotham, Star City); connect(Atlantis, Metropolis); connect(Central City, Coast City);

connect(Metropolis, Central City);

connect(Metropolis, Star City);

2 Dat-uhhh Structures

(a) How many nodes would be in the left subtree of a complete binary search tree where height (h) > 0 and the root starts at a height of 0. Note, if you want to use height in your calculations, you must use the height of the **full binary search tree rooted at the original root**, not subtrees.

(b) How many nodes would be in the right subtree of the right subtree in a full binary search tree (so the tree rooted at the right child of the root's right child). The same assumptions should be made as in the previous problem.

(c) What is the criteria that a heap must fulfill so that it can be a min heap and max heap at the same time.

(d) Can red-black trees have only black links (so no red links/nodes)? When is this the case?

(e) What modification would you have to do to a QuadTree to account for 3 dimensional nodes?

(f) What modification would you have to do to a K-D Tree to account for 3 dimensional nodes?

(g) What is wrong with the following code snippet? Assume that it compiles properly, there is a constructor, and a hashcode method that has a hashcode function works well (spreads everything well).

```
1 Public class Human{
2         int legs;
3         int arms;
4         @Override
5         public boolean equals(Human no) {
6         return legs == no.legs && arms == no.arms;
7         }
8         ....
9 }
```

3 Asymptotics

Write the Asymptotic runtime of the following function. Use the tightest bounds possible. Each problem is worth 4 points.

```
1  public static void mango(int N){
2          if(N<=1){
3                  return 1;
4          }
5     for int(i = 0; i < N / 2 ; i ++){
6          System.out.println('you');
7      }
8      mango(N - 1); mango(N-2);\\lie
9  }
```

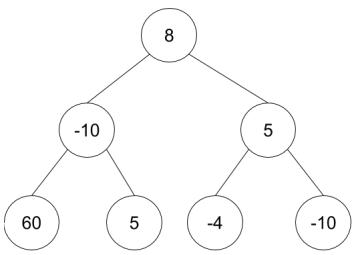
```
1  public static void stowb(int N){
2          if(N <= 1){
3                  return 1;
4          }
5          for(int i = 0; i < (N*1000)/N ; i ++){
6                  System.out.println("erry");
7          }
8          stowb(N/4);
9  }
```

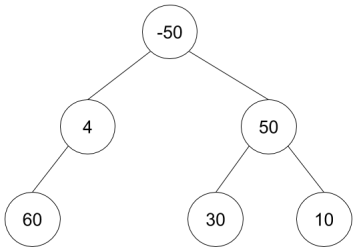
```
1  public static void grongrula(int N){
2          if(N<=1){
3                  return N;
4          }
5          for(int i = 0 ; i < N; i++){
6                  for (int j = 0 ; j < i; j++){
7                          System.out.print("Huh?");
8                  }
9          }
10         grongrula(N/2)
11 }
```

4 Arithmetic Trees

This problem deals with finding different types of paths inside of a Binary Tree. A path is defined as a set of nodes between any 2 nodes in the tree (This means the root may not be part of a path).

(a) To start off, we will attempt to find the maximum path sum in a Binary Tree. This means finding the path for which the sum is maximized. Your path must have at least one node but can have more. For example, the tree below on the left has a maximum path of 60+-10+8+5 = 63. On the right, we have a maximum path of 3+50+10 = 63





You are given 2 methods, a *pathsum* method, which returns the maximum path sum and a helper method *pathsumhelper* which aids in this endeavor.
Below is the api needed for a BSTNode

```
public class BSTNode{
    int val;
    BSTNode left, right;
    BSTNode(int d, BSTNode left, BSTNode right){
        val = d;
        this.left = left;
        this.right = right;
    }
}
```

Fill in the code below to finish finding the maxPathSum:

```
public class BST {
    private int max_so_far = Integer.MIN_VALUE;
    public int pathSum(BSTNode t){
        ------------------------------------------------

        ------------------------------------------------
    }

    private int maxPathSum(BSTNode t){
        if (t == null) {
            return 0;
        }
        ------------------------------------------------

        ------------------------------------------------

        ------------------------------------------------

        ------------------------------------------------

        return------------------------------------------
    }
}
```

(b) What is the runtime of your algorithm in part a in terms of N where N is the number of nodes in the Tree.

(c) Now say that instead of a normal Binary Tree, we were dealing with a **Binary Search Tree**. What optimization could we do to *MaxPathSum* to speed up our code? No need to write out code just write what you would do.

(d) What is the runtime of your algorithm in part c in terms of N where is the number of nodes in the Tree.

(e) Say we wanted to find the maximum product path in a normal **Binary Tree** where the maximum product path is defined as a path where the product of the numbers of each node in the path is its maximum value. What modification would you need to the algorithm for maximum product path in order to make this work?

(f) What constraints would we need to have on the nodes of the input **Binary Tree** such that you could take the above algorithm for finding the max path sum and replace all operations for addition with their corresponding multiplication ones and have it function without any other logic changes.

(g) Finally, say we wanted to find the maximum product path in a **Binary Search Tree**. What optimization could we do to speed it up? (Hint this combines parts c and g).

5 MashedMap

1. We have a HashMap, but a goon did something so that duplicates were not handled properly. This means that multiple versions of a key can exist in our HashMap. Luckily, your hashcode is still intact. In addition to this, the spread of the hashcode is pretty good and it distributes elements fairly evenly.

   Determine how long it would take in order to traverse your broken HashMap to find and delete ALL duplicates of ONE key. For example, how long would it take to find the key "macaroni". Provide the runtime (It is your choice what symbol to use) based off our prior assumptions. Give reasons for both runtimes. Correct runtimes with wrong explanations will be given 0 points.

2. You know for a fact that your HashMap class is perfect. You spent tireless hours in your 61B Lab making sure you had no flaws in your code of the HashMap. Assuming that your HashMap code is perfect how would this goon be able to ruin your duplicate handling given that he can only input items into your HashMap?

3. Pretend that you did not remove the duplicates from the faulty HashMap and you wanted to keep track of how many times each key was inside of the HashMap. How would one do this? We would like our solution to be as fast as possible.

6 Riddle Me This

What president wears the biggest hat?

7 Runtimes not Finished Call that Not Done Times

Write all runtimes in Theta if there is a tight bound, if there is not, write your runtime in terms of Omega and O. Assume nothing other than what is given in the problem.

a) Runtime of putting N presorted keys (with no duplicates) into a binary search tree.

b) You have a minheap and you will perform deleteMin operations until the min-heap is empty. Whenever an element is deleted, it is added to a Binary Search Tree.

c) Inserting N elements into a HashMap that all have the same key and different values.

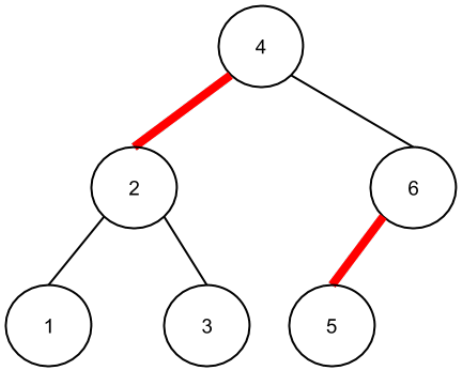
d) Given an array sorted from least to greatest (lowest index on left, highest index on right and no duplicates). Iterate through the array from left to right and at each index, insert the corresponding element into a stack. Finally, pop every element from the stack and insert it into a maxheap.

8 Rotations and Flips

a) What series of inserts would result in a perfectly balanced binary search tree for the following input: 1, 2, 3, 4, 5, 6 , 7 , 8 , 9 , 10, 11, 12, 13, 14, 15. Draw the resulting Binary Search Tree.

b) Delete the item 12 from the above Binary Search Tree and draw the results.

c) Insert 7 into the following red black tree. Show your steps using the red-black tree method for full credit. $\frac{1}{2}$ credit will be given for converting into a 2-3 tree and back. Denote red links with a dotted line.



d) How many rotations that will occur if you insert 5.5 into the new red black tree?

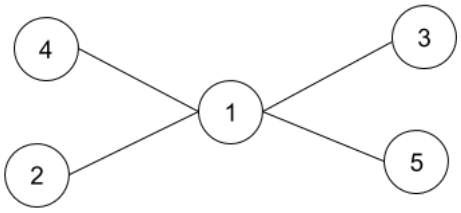
e) Give a brief description of an algorithm that you would use to get a minheap from a 2-3 Tree in $O(N)$ time where N is the amount of nodes in your tree.

f) We have a set of N words, in an alphabet with M characters, all of length L. However, we know that the $L-10$ letters are all the same (the last $L-1$ letters are all the same in the same order). How can we construct a Trie such that the following occurs

- Query to see if any word exists in our Trie takes constant time.
- Adding a word to our Trie, given it follows the same scheme as above takes constant time.

9 Graphical Blow to the Jaw

We have dealt with Breadth First Search in the class. Now we will go over another similar problem, bidirectional Breadth First Search. The jist of this problem is that we want to see if a path exists between 2 vertices in a graph. Instead of doing normal Breadth First Search, we choose to run 2 Breadth First Searches concurrently– one from the start vertex and one from the goal vertex. We then return the vertex that the searches intersect on. If the 2 vertices are not connected, we return -100. For example, in the below graph, we would return 1. Don't worry about accounting for odd lengths (they don't matter).



You will have to fill in 1 method to fill, *bidirbfs*, which takes in a source vertex s and a goal vertex g and returns the output of bidirectional BFS (the point at which the two BFS's meet). In addition to this, there is a helper method, *iteration* defined for you– cross it out if you choose not to use it.

Below are some methods for a graph that may be useful. Note all vertices are nonnegative consecutive integers in this problem (0,1,2,3.....N).

```
degree(int v); \\Returns the degree of some vertex v.
V(); \\Returns the number of vertices in a graph
E(); \\Returns the number of edges in the graph.
adj(v); \\Returns an iterable with all the vertices adjacent to v
addEdge(int v, int w); \\Adds an edge between vertices v and w
```

```
public class BiDirectionalBFS {
    Graph G;
    private class BFS {
        Queue<Integer> queue;
        boolean[] marked;

        BFS(int count, int source) {
            marked = new boolean[count];
            queue = new Queue<Integer>();
            queue.enqueue(source);
            marked[source] = true;
        }
    }
```

```
    private int iteration(_____) {

        _____

        for (_____) {

            if(_____){

                _____

                _____
            }

            if (_____) {

                _____
            }
        }
        return _____;
    }

    public int bidirbfs(int s, int g) {

         _____

         _____

        while (_____){

            _____

            _____

            if (_____){

            _____

            } else if(_____){

             _____

            }
        }
        return _____
    }
}
```

10 Josh's 61BBQ and Foot Massage

You are the Chef at a famous restaurant, *Josh's barbecue et Massage des Pieds*[1]. In your restaurant you have N possible menu items and you have M customers who come every day (you're a really good cook in this world). Each customer has a specific I.D number going from 1 to M.

You have a specific policy– once one customer has had one item from your menu, no one else can have it. To be fair, you have asked to customers to rank each of the N items with an integer from 1 to 100, where 1 is "61Bad" and 100 is "61Best". For every number between 1 and 100 a customer can rank $\frac{N}{100}$ items with that number; in other words there will be items that a customer ranks with the same number.

Every day, you will go through the customers in order of their I.D (from 1 to M) and create the order corresponding to the item which they rank the highest (as close to 100 as possible) that has not yet been picked. Once all menu items have been finished, you close your restaurant.

Your goal is design a system that does the following:

- Instantiating the preferences for all customers' preferences should take $O(MN)$ time
- Pick the item a customer should eat in $O(1)$ time
- After picking the item a customer should eat, it should be removed from everyone else's options in $O(M)$ time.

Use the below space to describe how you would use Data Structures we have learned in class (Arrays, LinkedLists, Stacks, Queues, Weighted Quick Union, Binary Search Tree, Priority Queues, Tries, Hashtables, 2-3 Trees, Red Black Trees, Quad Trees) to complete the problem.

[1] See the following link for more information on the restaurant https://tinyurl.com/JoshBBQ

11 Mini Algorithm Design

1. Give a brief description of an algorithm that you would use to get a minheap from a 2-3 Tree in $O(N)$ time where N is the amount of nodes in your tree.

2. We have a set of N words, in an alphabet with M characters, all of length L. However, we know that the $L - 10$ letters are all the same (the last $L - 10$ letters are all the same in the same order and we have knowledge of the first 10). How can we construct a Trie such that the following occurs

   - Query to see if any word exists in our Trie takes constant time.
   - Adding a word to our Trie, given it follows the same scheme as above takes constant time.