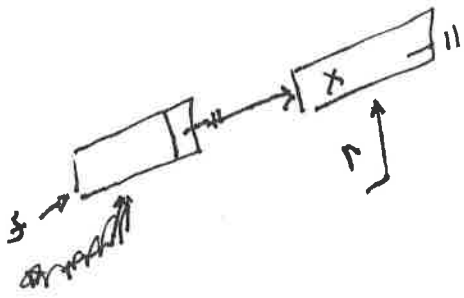
Print Full Name Key Andrew ID Key

Tree Problem - Code (15 points)

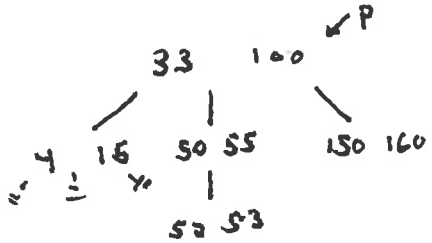
Consider the following class named TreeNode.java.

```
package midterm;
public class TreeNode {

   private int d1, d2;
   private TreeNode lc,mc, rc;

   public TreeNode(TreeNode lc, int d1, TreeNode mc, int d2, TreeNode rc){
      this.lc = lc;
      this.d1 = d1;
      this.mc = mc;
      this.d2 = d2;
      this.rc = rc;
   }

   @Override
   public String toString() {
      String LC = "";
      String RC = "";
      String MC = "";
      if (lc == null) LC = "||<-";
      else LC = "<--";
      if (mc == null) MC = "--";
      else MC = "|";
      if (rc == null) RC = "->||";
      else RC = "-->";
      String s = LC + d1 + MC + d2 + RC;
      return s;
   }

   public int getD1() {
      return d1;
   }
   public void setD1(int d1) {
      this.d1 = d1;
   }

   public int getD2() {
      return d2;
   }
```



```
public void setD2(int d2) {
   this.d2 = d2;
}

public TreeNode getLc() {
   return lc;
}
public void setLc(TreeNode lc) {
   this.lc = lc;
}

public TreeNode getRc() {
   return rc;
}

public void setRc(TreeNode rc) {
   this.rc = rc;
}
public TreeNode getMc() {
   return mc;
}
public void setMc(TreeNode mc) {
   this.mc = mc;
}

public static void main(String args[]) {
   TreeNode root = new TreeNode(null, 33, null, 100, null);
   System.out.println(root);
   TreeNode a = new TreeNode(null, 4, null, 15, null);
   root.setLc(a);
   TreeNode b = new TreeNode(null, 50, null, 55, null);
   root.setMc(b);
   TreeNode c = new TreeNode(null, 150, null, 160, null);
   root.setRc(c);
   TreeNode d = new TreeNode(null, 52, null, 53, null);
   b.setMc(d);
   TreeNode p = root;
   while(p != null) {
      System.out.println(p.getD1() + "   " + p.getD2());
      p = p.getMc();
   }
 }
}
```

Key



11 <- 33 -- 100 ->11

11 <- 33 -- 100 ->11

33 100
50 55
52 53

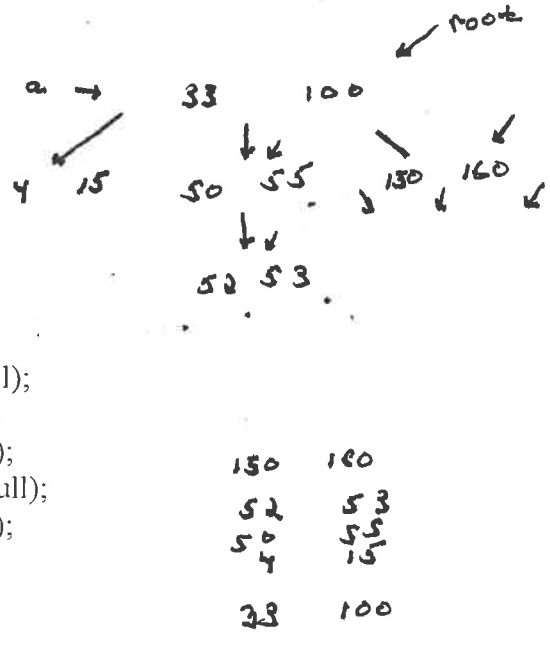
Key

1. There are four lines of output from TreeNode.java. Each line is printed with a println() statement. What is the exact output of these four lines?

11 <-> 33 <-> 100 -> 11 (1.25 points)
33 100 (1.25 points)
50 55 (1.25 points)
52 53 (1.25 points)

Consider the following class that uses the TreeNode from page 1. There are five lines of output from this program.

```
package midterm;
public class Tree {

   private TreeNode root;

   public Tree(TreeNode root) {
      this.root = root;
   }
   public void setRoot(TreeNode tree) {
      this.root = tree;
   }

   public TreeNode getRoot() {
      return root;
   }

   public void strangeTraversal(TreeNode t) {
      if(t != null) {
         strangeTraversal(t.getRc());
         strangeTraversal(t.getMc());
         strangeTraversal(t.getLc());
         System.out.println(t.getD1() +"  " +  t.getD2());
      }
   }
   public void strangeTraversal(){
      strangeTraversal(root);
   }
   public static void main(String[] args) {
      TreeNode a = new TreeNode(null, 33, null, 100, null);
      TreeNode b = new TreeNode(null, 4, null, 15, null);
      TreeNode c = new TreeNode(null, 50, null, 55, null);
      TreeNode d = new TreeNode(null, 150, null, 160, null);
      TreeNode e = new TreeNode(null, 52, null, 53, null);
      Tree t = new Tree(a);
      t.getRoot().setLc(b);
      t.getRoot().setRc(d);
```



150 160
52 53
50 55
4 15
33 100

```
        t.getRoot().setMc(c);
        t.getRoot().getMc().setMc(e);
        t.strangeTraversal();
    }
}
```

Key 1

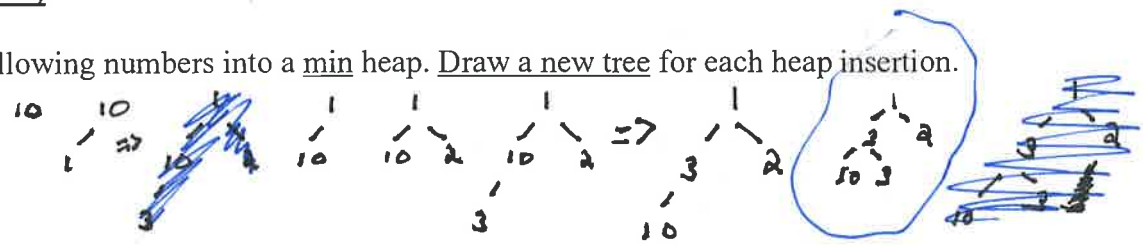
2) Show the five lines of output here. (2 Points per line = 10 points)

150	160
52	53
50	55
4	15
33	100

Heaps (16 points)

3) Insert the following numbers into a min heap. Draw a new tree for each heap insertion. (6 Points)

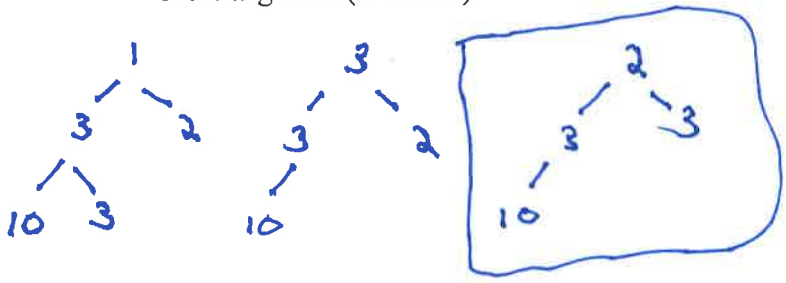
10,1,2,3,3

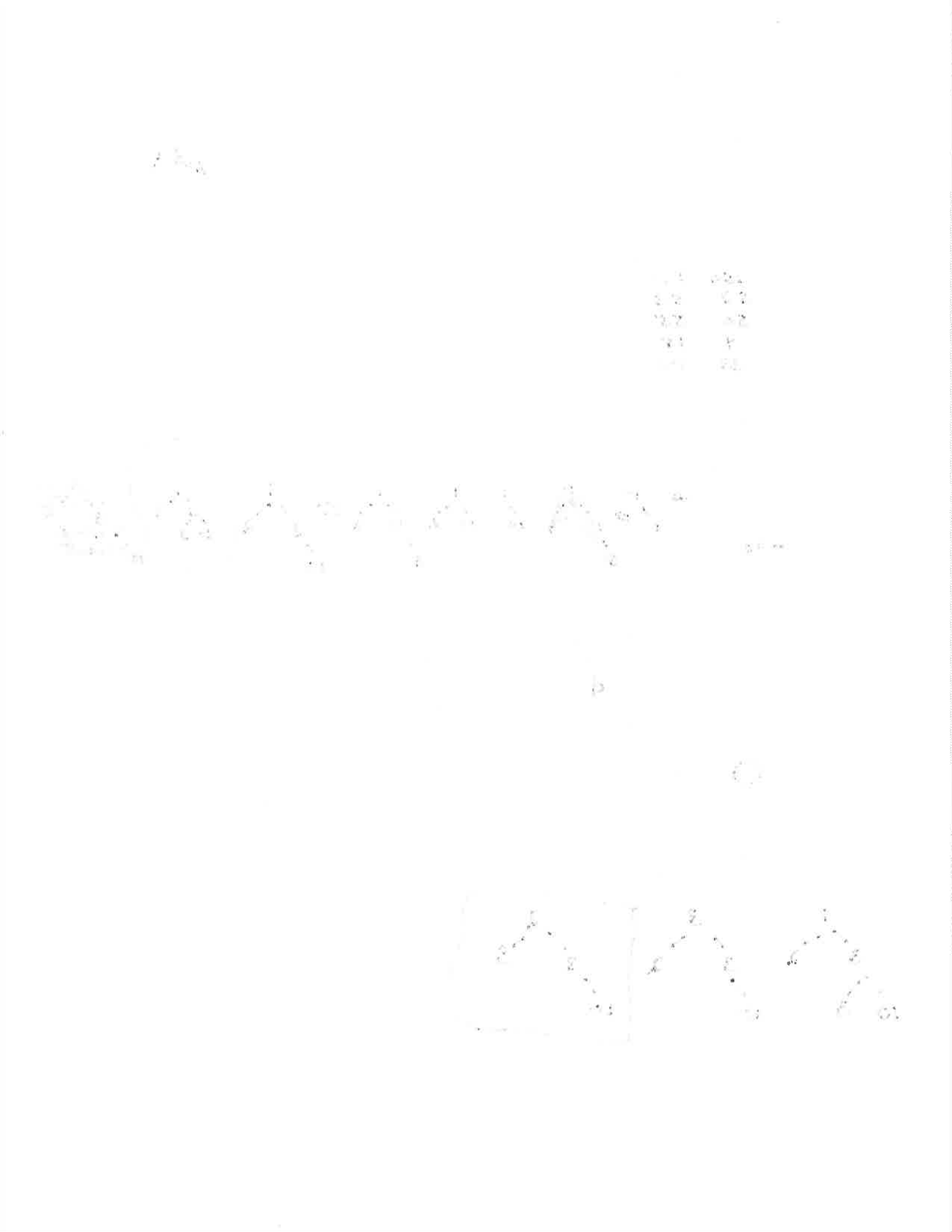


4) Suppose we were to add three more numbers to the heap in question 3. Which of the following is correct? d (2 Points)

a. The new height would be 2^n, where n is the number of numbers in the heap.
b. We cannot determine the new height. It depends on the values of the numbers.
c. The new height would be two.
d. The new height would be three.
e. The new height would be eight.

5) Perform a single delete operation on the heap that you drew in question 3. Draw the resulting tree. (3 Points)

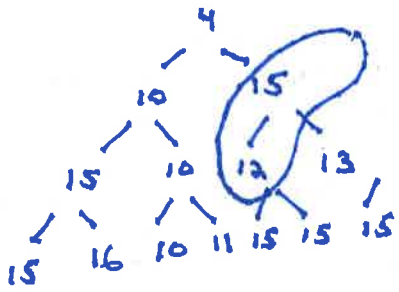




Keys

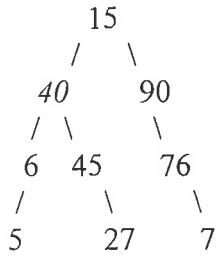
6) Consider the following min heap implemented in an array. It is not quite correct. To make it a min heap exactly one swap of parent and child must occur. What two numbers need to be swapped in order to make this a min heap? (5 points). Circle the two numbers.

4
10
15
15
10
12
13
15
16
10
11
15
15
15



Binary Trees (16 points)

7. Parts (a), (b), and (c) refer to the following binary tree:



(a) List the data that would be accessed by a pre-order traversal on the given tree by writing out the values in the nodes as they would be accessed, separated by commas. (3 points)

15, 40, 6, 5, 45, 27, 90, 76, 7

(b) List the data that would be accessed by an in-order traversal on the given tree by writing out the values in the nodes as they would be accessed, separated by commas. (2 points)

5, 6, 40, 45, 27, 15, 90, 76, 7

Key 6

(c) List the data that would be accessed by a post-order traversal on the given tree by writing out the values in the nodes as they would be accessed, separated by commas. (2 points)

5, 6, 27, 45, 40, 7, 76, 90, 15

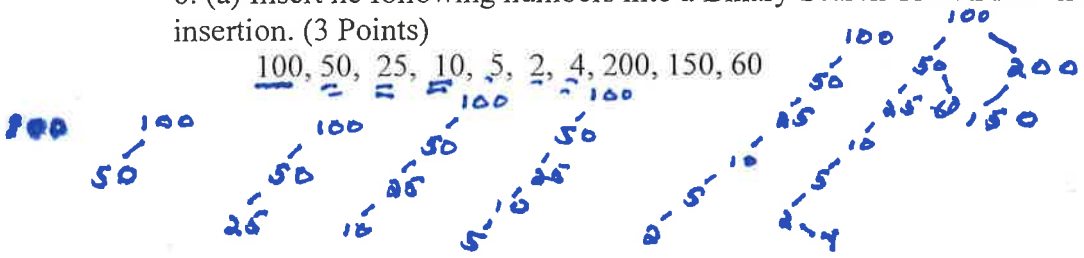
(d) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and complete with height h, how many leaves, in terms of h, will the tree have? (2 points) 2^h Note, this tree has a perfectly flat bottom.

(e) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and complete with exactly k leaves. What is the height (in terms of k) of this tree? (2 points) $\log_2 K$ Note, this tree has a perfectly flat bottom.

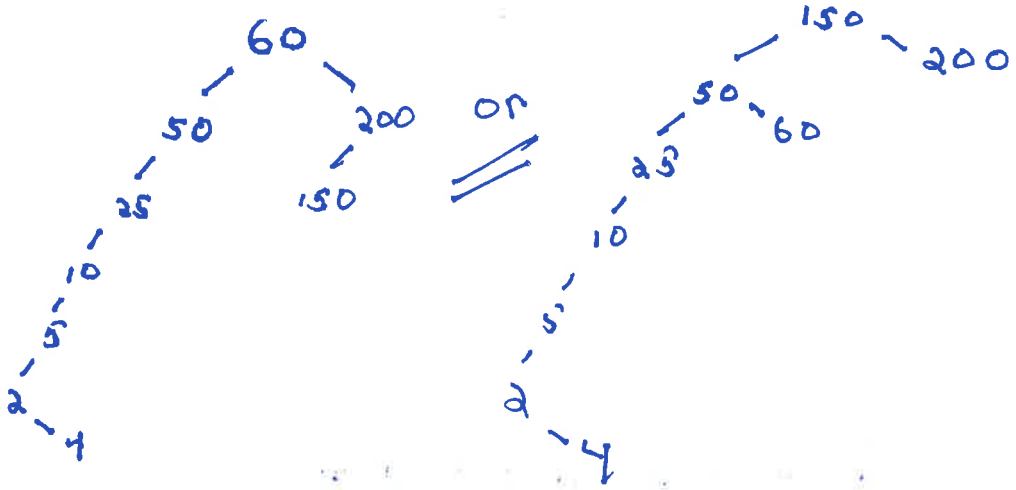
ON board => show Final tree

8. (a) Insert he following numbers into a Binary Search Tree. Draw the tree after each insertion. (3 Points)

100, 50, 25, 10, 5, 2, 4, 200, 150, 60



(b) Delete 100 from the final tree that you drew in 8 (a). We discussed two ways to do this. You must perform the deletion in one of these two ways. Draw this final tree. (2 Points)



Key 1

Project Questions (18 points)

9. Recall the Merkle-Hellman cryptosystem and the Merkle tree that we worked with in Project 1, the spell checker application and the dynamic programming exercise in Project 2, and the calculator problem from Project 3.

The Merkle-Hellman cryptosystem in Project 1 was based on the subset sum problem which is known to be NP-Complete. The problem itself can be described as follows: given a set of numbers *X* and a number k, is there a subset of *X*, which sums to k?

(a) Suppose X = {4, 16, 3, 9, 2, 8, 5} and k = 31. Is there a subset of X which sums to k?
Yes Yes/No (2 points)

(b) Suppose Alice sends messages to Bob encrypted with Bob's Merkle-Hellman public key. Circle the one statement that is true? (2 Points)

1. Bob decrypts with his public key.
2. Alice encrypts with a super increasing sequence.
3. Alice decrypts with a super increasing sequence.
4. Bob decrypts with a set such as X in part a.
5. Alice encrypts with a set such as X in part a.

(c) Write a method in Java that returns true if there is a subset of X which sums to k and false otherwise. The method signature and pre-conditions are provided. Note, the array x is a super increasing sequence. (6 points)

```
public static boolean subSetSum(int[] x, int n, int k) {
// pre: x is a super increasing and x[0] < x[1] <x[2] <...<x[n-1].
// pre: n is the size of x.
// pre: all integers involved are small enough that overflow is not of concern
// post: returns true if some subset of x sums to k, false otherwise.
int i = n-1
while (i >= 0) {
    if (x[i] <= k) k = k - x[i].
    if k == 0 return true;
} i = i - 1;
return false;
```

i i i k
1 2 4 8 16
i i i
4 8 13
k
2
k
16
2
k
31
15
0

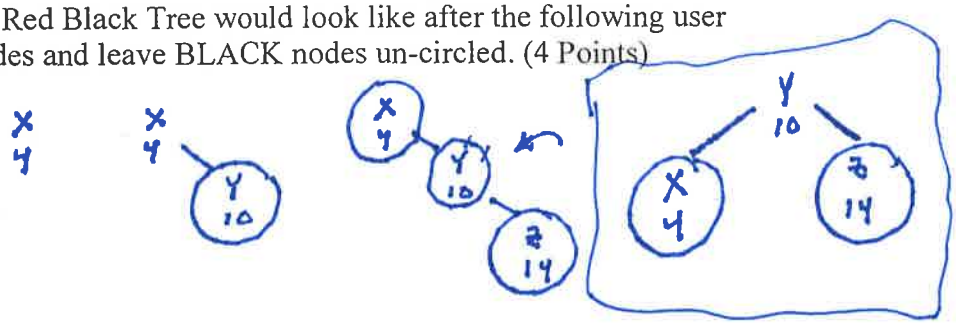
(d) What is the best case run time complexity of your code in part (c)? Use Big Theta notation. (1 Points) Θ(1)

Key B

(e) In project 1 we wrote a recursive function that computed probabilities of a World Series win. Briefly describe the run time performance of this function. (1 Point) it blew up exponentially.

(f) In Project 3, we wrote a calculator that processed RPN expressions and used a Red Black Tree. Draw what the Red Black Tree would look like after the following user interaction. Circle RED nodes and leave BLACK nodes un-circled. (4 Points)

```
X 4 =
Y 10 =
Z X Y + =
```



(g) In Project 2 we wrote a spell checker that loaded n words into a Red Black tree and allowed a user to make queries against the tree. We wrote a method named lookup() that checked if a word was present. Which of the following is true of the worst-case lookup method? Circle all correct answers. (2 Points)

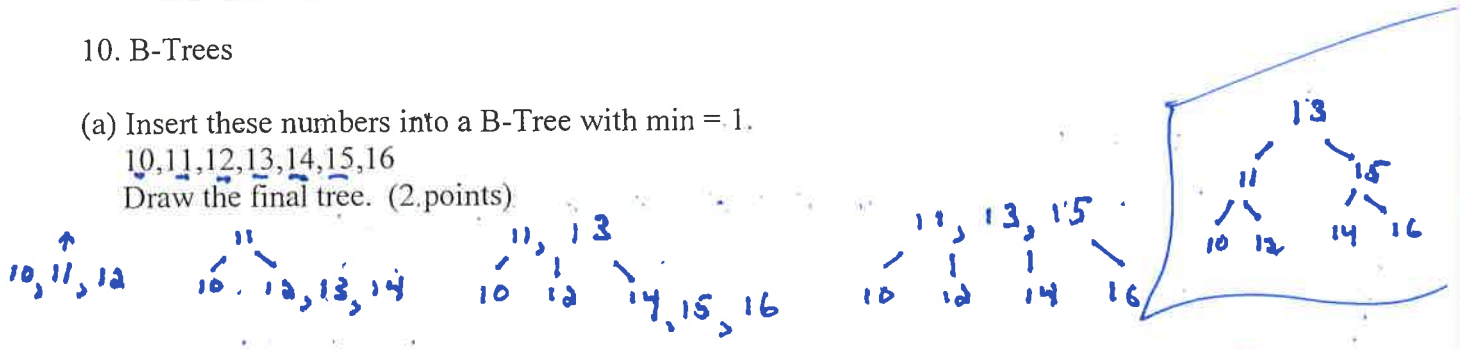
1. It ran in O(LogN)
2. It ran in O(1)
3. It ran in Ω(N^2)
4. It ran in Θ(N)
5. It ran in Θ(LogN)
6. It ran in Θ(1)
7. It ran in O(2^N)
8. It ran in O(3^N)

-1 for each wrong. max off is 2.

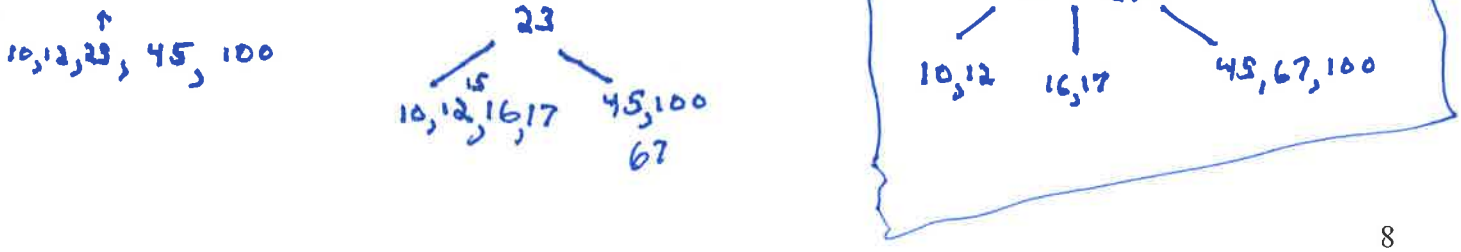
Balanced Trees (15 points)

10. B-Trees

(a) Insert these numbers into a B-Tree with min = 1.
10,11,12,13,14,15,16
Draw the final tree. (2 points)



(b) Insert the numbers 45, 23, 12, 10, 100, 16, 17, 67, 15 into a B-Tree with min = 2. (2 points)

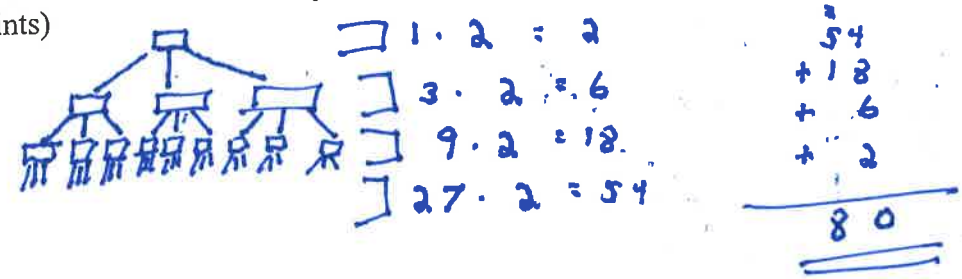


Key 9

(c) What is the height of the B-Tree in question 10 (a)? 2 (1 Point)

(d) What is the height of the B-Tree in question 10 (b)? 1 (1 Point)

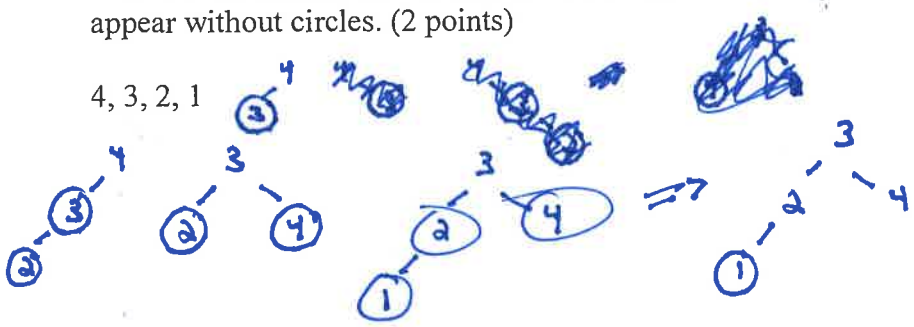
(e) What is the maximum number of keys a B-Tree can hold with min = 1 and height = 3?
80 (2 Points)



11. Red-Black Trees

(a) Insert the following numbers, one by one, into a Red-Black Tree. Show the tree after each insertion. red vertices should be circled and black vertices should appear without circles. (2 points)

4, 3, 2, 1

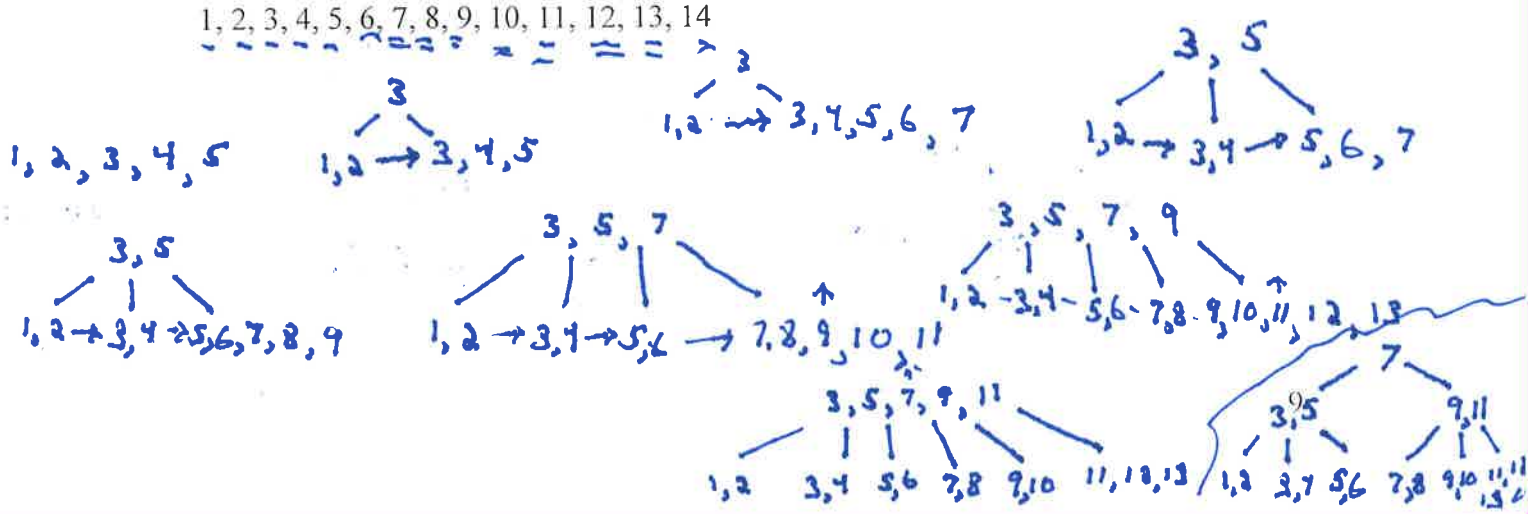


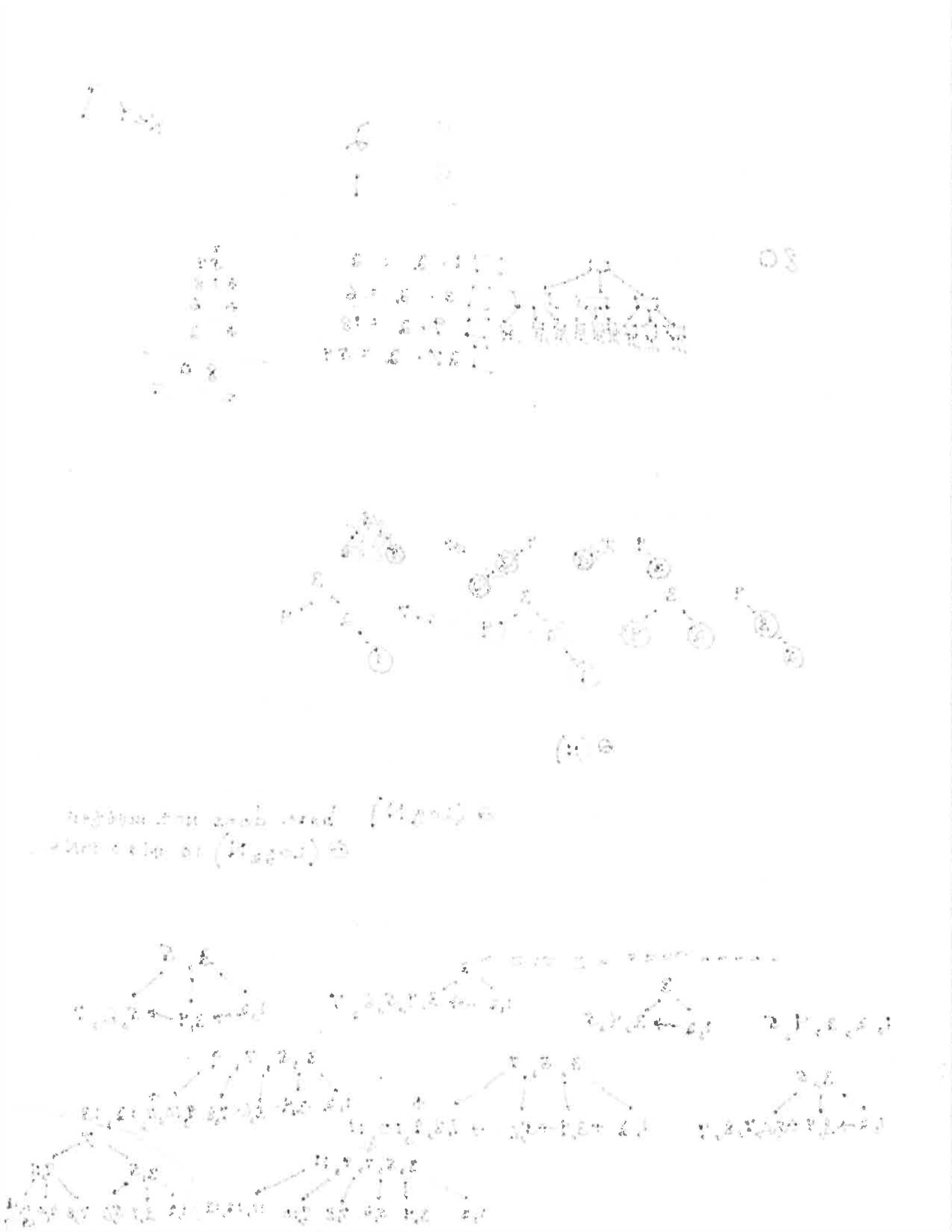
(b) What is the runtime complexity of an inorder traversal of a Red Black Tree ? Use Big Theta notation. (1 Point) $\Theta(N)$

(c) What is the worst-case runtime complexity of a Red Black Tree lookup operation? Use Big Theta notation. (1 point) $\Theta(\log N)$ base does not matter $\Theta(\log_2 N)$ is also ANS.

12. B+ Trees

Insert the following numbers into a B+ Tree with minimum = 2. (3 Points)
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

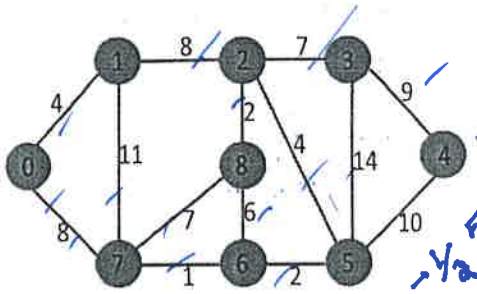


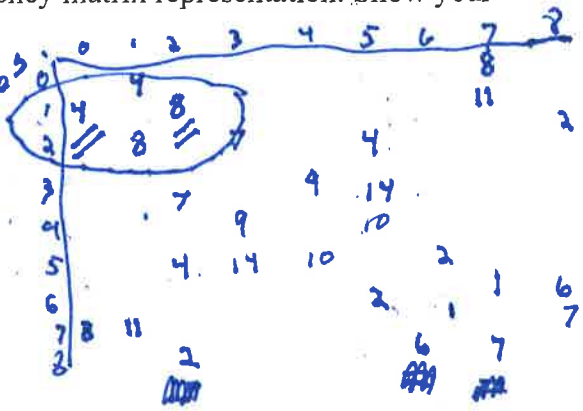


Key 10

Graph Representations and Concepts (8 points)

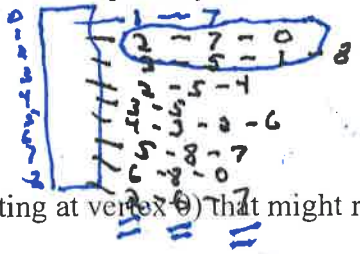
13. (a) Represent the following graph in an adjacency matrix representation. Show your matrix. (1 Points)





-½ for no distances

13. (b) Represent the graph above as an adjacency list. Draw a sketch of this representation. (1 Points)



not grading distances.

13. (c) Give a list of vertices (starting at vertex 0) that might result from a breadth first traversal of this graph. (2 Points)

0, {1, 7}, {2, 8, 6}, {3, 5}, 4

13. (d) Give a list of vertices (starting at vertex 0) that might result from a depth first traversal of this graph. (2 Points)

0, 1, 2, 3, 4, 5, 6, 7, 8

13. (e) A Hamiltonian cycle is a closed loop through a graph that visits each node exactly once. Does the graph above contain a Hamiltonian cycle? Circle Yes or No (2 Points)

0, 1, 2, 3, 4, 5, 6, 8, 7, 0

Key 11

Queue (3 points)

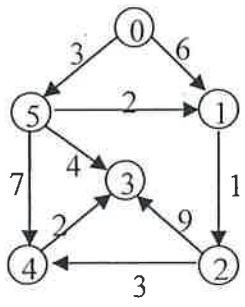
14. Write a class Queue. The queue data will live in a linked list. The queue will hold simple integers. You will provide a constructor, a method named addToQueue, a method named removeFromQueue, and a method named isEmpty. This queue must be written in Java and the syntax must be good but not perfect. For full credit, be exceptionally neat. Your addToQueue and your removeFromQueue operations must run in $\Theta(1)$ time.

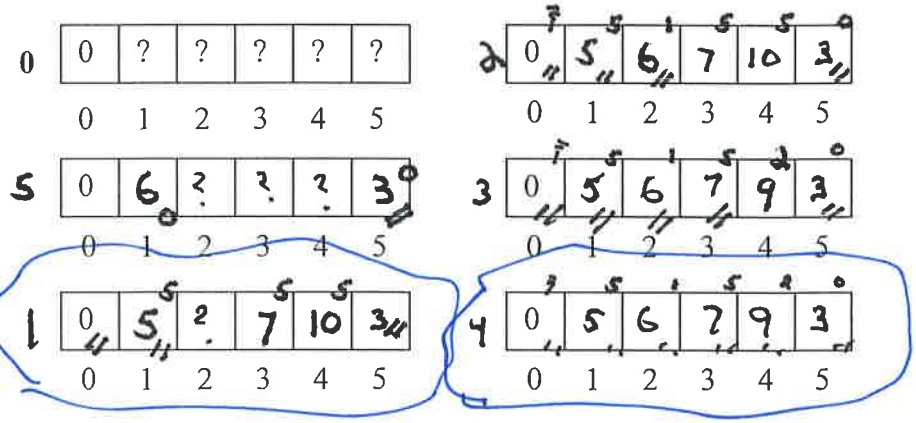
```
class Node {
    int data;
    Node next;
    Node(int d, Node N) {
        data = d;
        next = N;
    }
    int getData() {
        return data;
    }
    void getData(int x) {
        data = x;
    }
    Node getNext() {
        return next;
    }
    void setNext(Node N) {
        next = N;
    }
}

class Queue {
    Node f;
    Node r;
    Queue() {
        f = null; r = null;
    }
    void addToQueue(int x) {
        if (f == null) {
            f = new Node(x, null);
            r = f;
        }
        else {
            r.next = new Node(x, null);
            r = r.next;
        }
    }

    boolean isEmpty() {
        return (f == null);
    }

    // PRE: queue is not empty
    int removeFromQueue() {
        int t = f.data;
        f = f.getNext();
        return t;
    }
}
```

Key 12

Dijkstra's Shortest Path (9 Points)

15. Draw the contents of the distance array for each iteration of Dijkstra's Algorithm as it works on the graph sketched here. The initial state is given. **Mark the node to be selected next to the left of the array (note how 0 is marked to the left of the first array.)** Fill in each array cell working downward. That is, complete the first column of arrays before the second column of arrays. (6 Points)





(d) After Dijkstra is complete it also collects parent pointers for each node in the graph. Complete the chart below showing the parent of each node as computed by Dijkstra. (3 point)

Node	Parent
0	Nil
1	5
2	1
3	5
4	2
5	0