Lists (15 points)

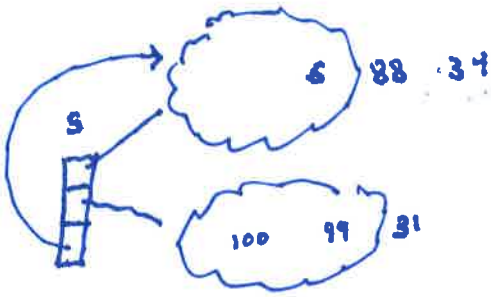
Name KEY

1. Consider the following Node and List classes.

```
class Node {

    private Object data;
    private Node next;

    public  Node(Object d, Node n) { data = d; next = n; }

    public  Node getNext() { return next; }

    public  Object getData() { return data; }

    public  void setNext(Node n) { next = n; }

    public  void setData(Object x) { data = x; }

    public  String toString() {

        String suffix = "";
        if(next == null)
            suffix = "--|";
        else   suffix = "-->";
        return data.toString() + suffix;
    }
}

public class List {

    private Node head;

    public List() { head = null;}

    public void add(Object t) { head = new Node(t,head); }

    public void addAlso(Object t) {
        if (head == null) {
            head = new Node(t,null);
        }
        else {
            Node p = head;
            Node q = p;
            while(p != null) {
                q = p;
                p = p.getNext();
            }
            Node m = new Node(t,null);
            q.setNext(m);
        }
    }


    public Object remove() {  Object t = head;
        head = head.getNext();
        return t;
    }
```

Key 2

```
    public boolean isEmpty() { return (head == null); }

    public String toString() {

        if(head == null) return "--|";

        String result = "";

        Node p = head;

        while(p != null) {

            result = result + p;

            p = p.getNext();
        }

        return result;
    }

    public static void main(String a[]) {

        List s[] = new List[3];

        s[0] = new List();
        s[1] = new List();

        s[0].addAlso(5);
        s[0].addAlso(88);
        s[0].addAlso(34);

        s[1].add(31);
        s[1].add(99);
        s[1].add(100);

        // 1
        System.out.println(s[1]);

        // 2
        System.out.println(s[0]);

        //3
        while(!s[0].isEmpty())
            System.out.print(s[0].remove());

        //4
        s[2] = s[0];
        while(!s[2].isEmpty())
            System.out.print(s[2].remove());

    }

}
```



① 100 --> 99 --> 31 --|

② 5 --> 88 --> 34 --|

③ 5 --> 88 --> 34 --|

④ Nothing
s[2] is empty.

(a) The program compiles and executes without error. What is the exact output of the println statement marked in the code with a 1? (2 Points)

100 --> 99 --> 31 --|

~~5 --> 88 --> 34 --|~~

Key 3

(b) The program compiles and executes without error. What is the exact output of the while loop marked in the code with a 2? (2 Points)

5 --> 88 --> 34 --|

(c) The program compiles and executes without error. What is the exact output of the println statement marked in the code with a 3? (2 Points)

5 --> 88 --> 34 --|

(d) The program compiles and executes without error. What is the exact output of the println statement marked in the code with a 4? (2 Points)

No Output

(e) State a pre-condition that should be present in the remove method of the List class. (2 Point)

head != null or list not empty

(f) We need a public method for the List class that displays the list in reverse order. This method will be called "displayReversed" and is shown below. It will make use of a recursive method called "reverse" with the signature as shown. Complete the recursive method called "reverse". (5 Points)

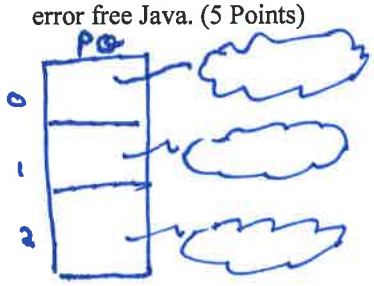
```
private void reverse(Node p) {
    IF (P == NULL) RETURN;
    ELSE {  REVERSE(P.GETNEXT());
            SYSTEM.OUT.PRINTLN(P);
         }
}

public void displayReversed() {
  reverse(head);
}
```

没写 没写判断 null -2.

Key

Priority Queues: (15 points)

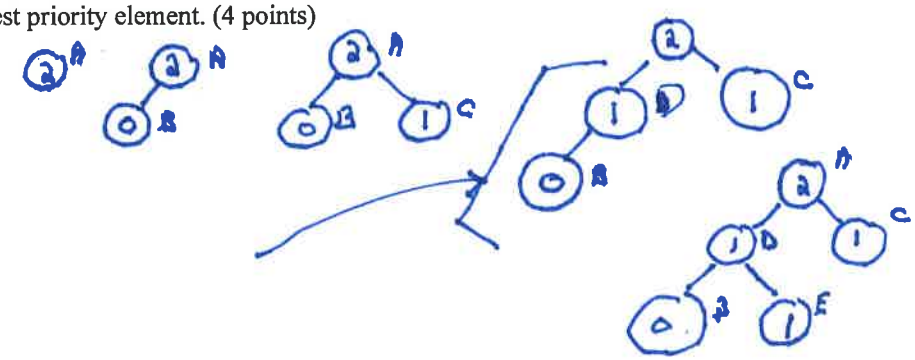
2) (a) Sue has implemented a priority queue as an array of linked lists. Each list has an enQueue and deQueue method and these methods run in θ(1). The enQueue method adds data at the rear of the queue and the deQueue method removes data from the front of the queue. Each array cell holds a pointer to the front of the queue and a pointer to the rear of the queue. There are 3 priorities and so the array has 3 elements. The highest priority is 2 and the lowest is 0. Provide an algorithm that shows how the highest priority element will be removed from such a queue. Your algorithm will find the highest priority element and remove it by calling deQueue on the first non-empty linked list. The code that you write needs to be syntax error free Java. (5 Points)




```
J = 2;
while (J >= 0) {
    IF (!PQ[J].isEmpty()) return PQ[J].dequeue
    J--;
}
return null;
```

(b) Bill has implemented a priority queue as a heap in an array. Draw a picture of the heap as a tree as it is being built. There are 3 priorities. The highest priority is 2 and the lowest is 0. Show each node being entered by redrawing the entire tree after each of the following insertions. Of course, Bill wants a deletion to always return the highest priority element. (4 points)

Ms. A has priority 2
Mr. B has priority 0
Ms. C has priority 1
Mr. D has priority 1
Ms. E has priority 1

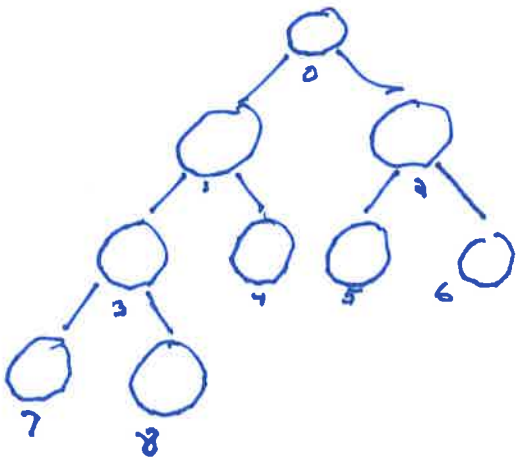


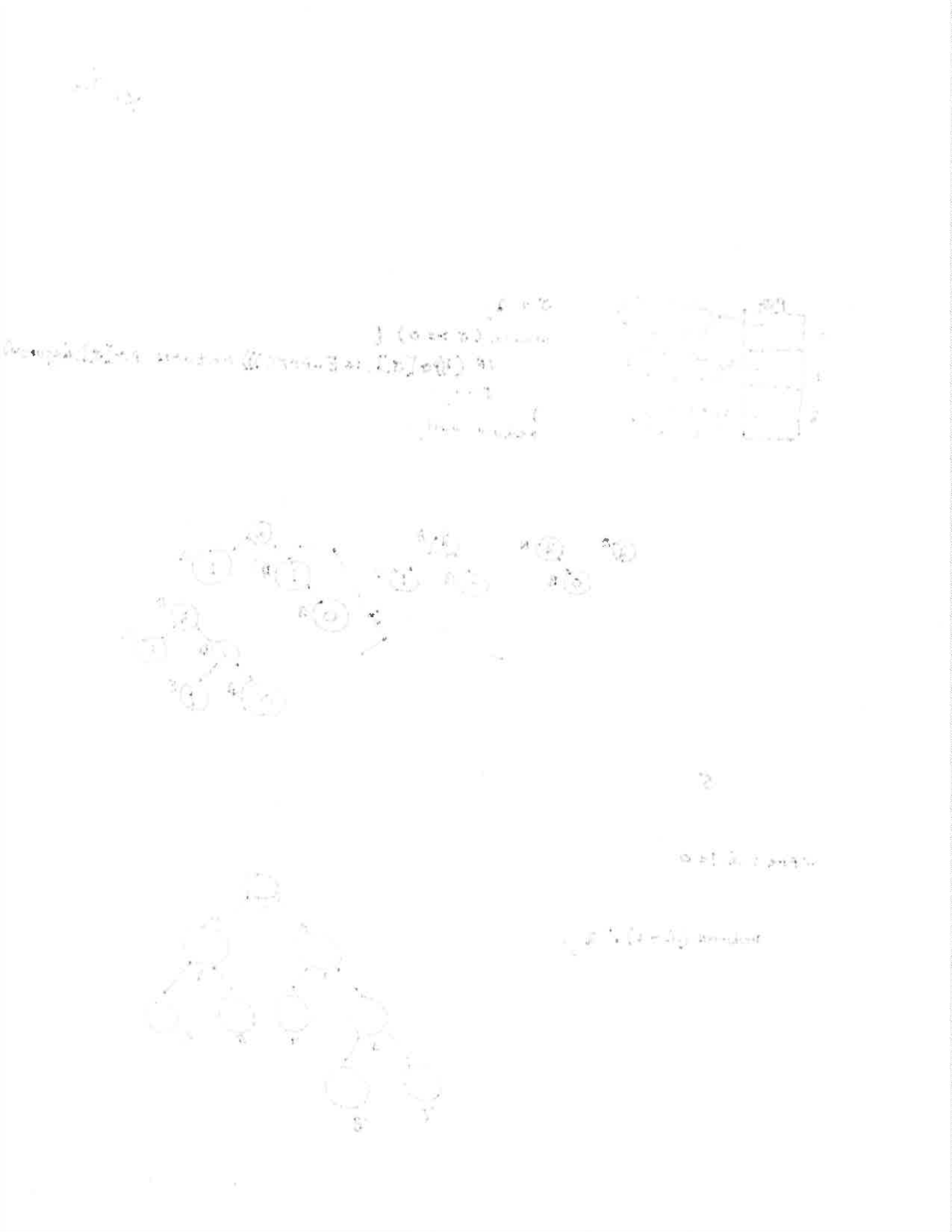
(c) How many array elements would the heap in question 2b occupy? (3 points)

5

(d) Suppose i is an array index into the heap in question 2b, complete the following method in error free Java. Be sure to express any pre-conditions that might be required.(3 Points)

```
// pre : i != 0
int parent(int i) {

    return (i - 1) / 2;

}
```

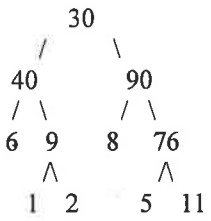




Keys

Trees (15 points)

3. Parts (a), (b), and (c) refer to the 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)

30, 40, 6, 9, 1, 2, 90, 8, 76, 5, 11

(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)

6, 40, 1, 9, 2, 30, 8, 90, 5, 76, 11

(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)

6, 1, 2, 9, 40, 8, 5, 11, 76, 90, 30

(d) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and the tree is of height h then how many leaves will the tree have? (4 points) 2^h

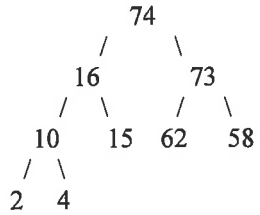
(e) Suppose we are working with a tree that has 0 or 4 children per node. Suppose too that the tree is perfectly balanced and is of height h. how many leaves will the tree have? (4 points)

4^h

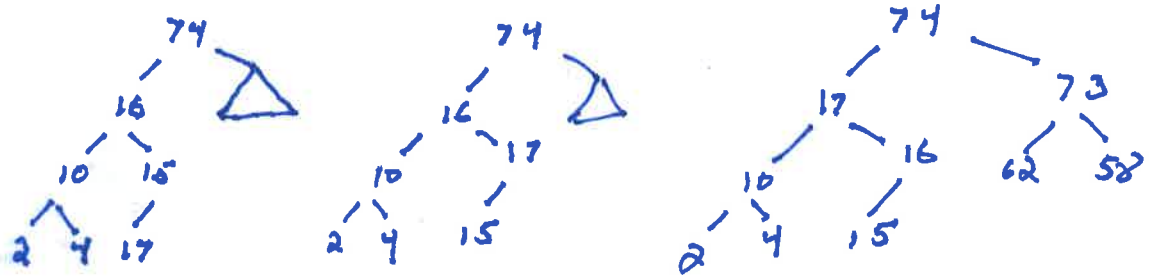
Key 6

Heaps (10 points)

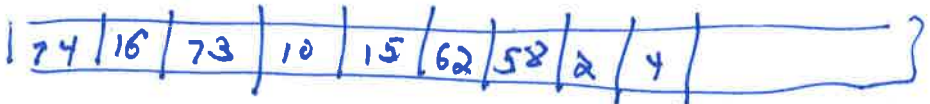
4. Consider the heap given below:



(a) Show the series of steps that would take place in adding a node containing 17 to the heap given above. In your answer, you should show what the heap looks like after <u>each step</u> in the addition process or reheapification process so that it is clear to the reader how the changes were made. (4 points)



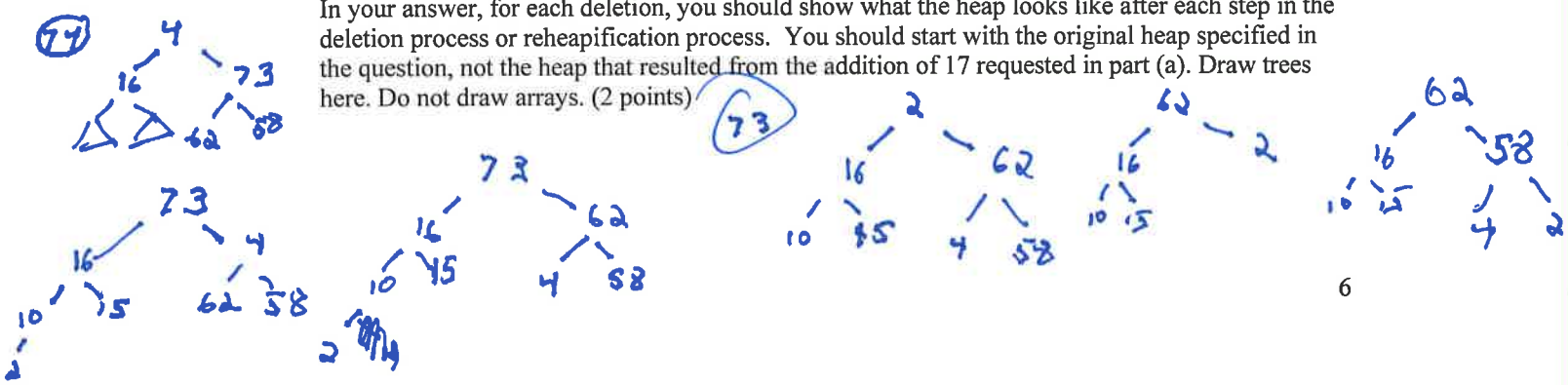
(b) Draw the heap in question 4 a. in an array. Be sure to show the array indexes. Draw the heap before the 17 was added to the heap in question 4 (a). (2 points)



(c) Consider again the heap in question 4 a. (without the 17). Describe a best case insertion into this heap. In other words, <u>specify a</u> value that you would add to this queue that would produce a best case insertion and specify the big θ performance of this insertion. (2 points)

value ≤ 15 big theta Θ(1)

(d) Show the series of steps that would take place in deleting two values from the heap shown above. In your answer, for each deletion, you should show what the heap looks like after each step in the deletion process or reheapification process. You should start with the original heap specified in the question, not the heap that resulted from the addition of 17 requested in part (a). Draw trees here. Do not draw arrays. (2 points)



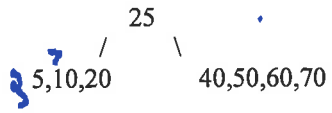
95-771 Data Structures and Algorithms
Spring 2020

Midterm Exam

Key

B-Trees (20 points)

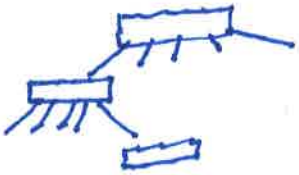
5. Consider the following B-Tree with a minimum of two and a maximum of four.



(a) Redraw the tree after inserting 2 and then 7. (3 points)

(b) Redraw the tree after inserting 42. (Begin work from the initial tree in question (5), do not work from the tree that you drew in question 5 (a).) (3 points)

(c) The original tree in question 5 has a height of 1. What is the maximum number of keys that this type of tree could hold with a height of 2? (3 points).



$4 \cdot 1 = 4$

$4 \cdot 5 = 20$

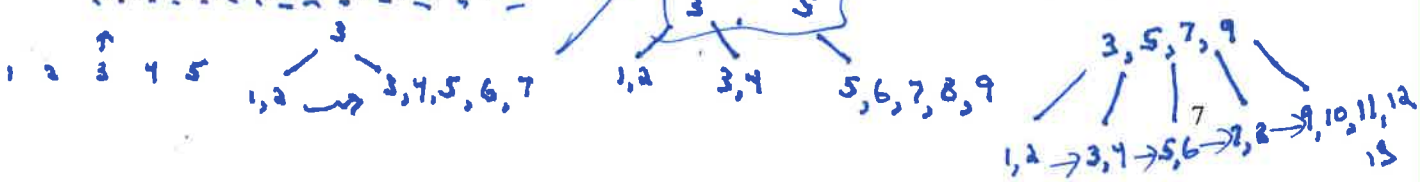
$+ \quad 4 \cdot 5 \cdot 5 = 100$

124

(d) The following tree has a minimum of 1 and a maximum of 2. Delete the number 60 from this tree and redraw the tree.(3 Points)

(e) The following tree has a minimum of 1 and a maximum of 2. Delete the number 70 from this tree and redraw the tree.(3 Points)

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

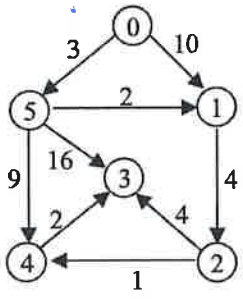


Key 8

Graph Algorithms(15 points)

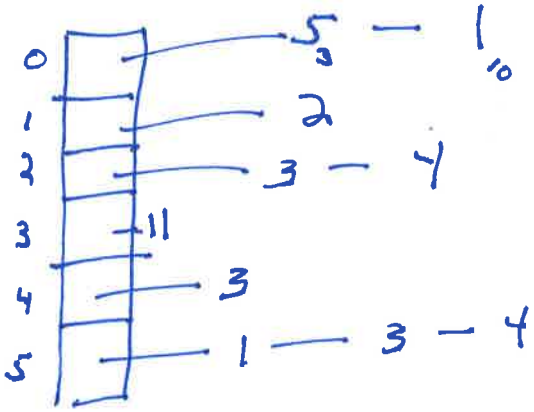
6. (a) What is the distance of the shortest path from node 0 to node 3 in the graph immediately below? (1 point) 12

(b) Draw the contents of the `distance` array for each iteration of Dijkstra's Algorithm as it works on this graph. Note how node 0 was selected and is shown to the left of the distance array. You need to show how the other nodes are selected in turn. Note the little blank "_" character next to each distance array. Fill that blank in with the next appropriate vertex number. Complete the first column first, and the next column second. (6 Points)

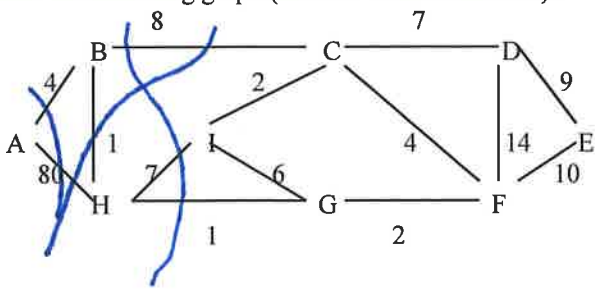


	0	1	2	3	4	5
0	0	?	?	?	?	?
2	0	5	9	19	12	3
5	0	10	?	?	?	3
4	0	5	9	13	10	3
1	0	5	?	19	12	3
3	0	5	9	12	10	3

(c) Draw an adjacency list representation of the graph shown immediately above (in part b). (3 Points)



(d) Consider the following graph (the root vertex will be A):



Key 9

Prim's algorithm first initializes the key and parent arrays in such a way that the root vertex has key 0 and a nil parent and all of the other vertices have an infinite key (?). It then continues to make selections and make updates to the key and parent arrays. Show the values that Prim's algorithm would compute for the second, third, and fourth pair of arrays below. (5 points)

	A	B	C	D	E	F	G	H	I
A	0	4	?	?	?	?	?	80	?
	nil	A						A	
B	0	?4	?8	?	?	?	?	?1	?
	nil	A	B					B	
H	0	?4	?8	?	?	?	?1	?1	?7
	Nil	A	B				H	B	H
G	0	?4	?8	?	?	?2	?1	?1	?6
	Nil	A	B			G	H	B	G

key 10

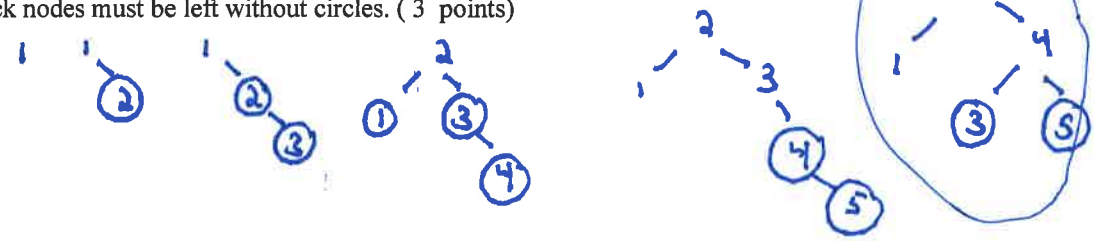
7. Project Questions (10 Points)

(a) In Project 2 we wrote a spell checker that loaded N words into a balanced Red Black tree and allowed a user to make queries against the tree. Suppose we wrote a method called traverse() that simply displayed each word in the tree. Suppose too that traverse() was carefully written and did not waste time. Which of the following is true of the traverse() method? Circle all correct answers. (3 Points)

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

(b) Enter the following numbers into a Red Black Tree and draw the final tree. Red nodes must be circled and Black nodes must be left without circles. (3 points)

1, 2, 3, 4, 5



(c) Write an algorithm (this does not need to be in error free Java) that correctly evaluates a single postfix expression. It leaves the correct answer on the top of the stack. You may assume that you have data structures such as stacks and queues and linked lists available. There are no variables in the expression. The only tokens are integer constants and the binary operators "+" and "*". Your algorithm should be exceptionally neat and easy to read and understand. It reads a single line of input and writes a single line of output. The output line pops the correct answer off the top of the stack. You may assume that you can write code such as :

for each token do {

}

Write the complete algorithm here (4 Points) :

readline
for each token do
    IF operand PUSH IT
    IF operator
        X = POP
        Y = POP
        APPLY OPERATION TO OPERANDS
        AND PUSH RESULT

PRINT POP