95-771 Data Structures and Algorithms
Fall 2021

Midterm Exam
Carnegie Mellon University

Key.

Name Key **Andrew ID** Key

Total points = 131. Score will be a percentage of 131.

Tracing Lists and Trees (28 points)

1. You will be asked to show the exact output of the following program. (18 Points)

```
package f21midterm;

class Node {
    private int data;
    private Node next;

    public int getData() {
        return data;
    }

    public Node getNext() {
        return next;
    }

    public void setData(int data) {
        this.data = data;
    }

    public void setNext(Node next) {
        this.next = next;
    }

    public Node(int data, Node next) {
        this.data = data;
        this.next = next;
    }
}
class List {
    Node head;
    Node tail;
    public List() {
        head = null;
        tail = null;
    }
    public void add(int x) {
        if(head == null) {
            head = new Node(x,head);
            tail = head;
        }
        else {
            tail.setNext(new Node(x,null));
            tail = tail.getNext();
        }
    }

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

```
    // (1.a) Pre: LIST IS NOT EMPTY
    public int remove() {
        int x = head.getData();
        head = head.getNext();
        return x;
    }

    // (1.b) Big Theta: Θ(N) where N is the size of the LIST
    public String toString() {
        Node v = head;
        String s = "";
        while(v != null) {
            s = s + v.getData() + " ";
            v = v.getNext();
        }
        return s;
    }
    // (1.c) Big Theta: Θ(N) where N is the size of the LIST
    public void foo() {
        tail = head;
        Node currNode = head;
        Node nextNode = null;
        Node prevNode = null;
        while(currNode != null) {
            nextNode = currNode.getNext();
            currNode.setNext(prevNode);
            prevNode = currNode;
            currNode = nextNode;
        }
        head = prevNode;
    }

}

public class DSMidterm {

    public static void main(String[] args) {
        List a = new List();
        a.add(1);
        a.add(2);
        a.add(3);
        a.add(4);
        System.out.println(a);    // (1.d) Output of println
        while(!a.isEmpty()) {
            System.out.print(a.remove() + "  ");
        }   // (1.e) Output of all the while() print
        a.add(1);
        a.add(2);
        a.add(3);
        a.add(4);
        a.foo();
        a.foo();
        a.foo();
        a.add(100);
        System.out.println(a);  // (1.f ) Output of println
    }
}
```

1 2 3 4

1 2 3 4

1 - 2 - 3 - 4

4 - 3 - 2 - 1

1 - 2 - 3 - 4

4 - 3 - 2 - 1

4 - 3 - 2 - 1 - 100

4 3 2 1 100

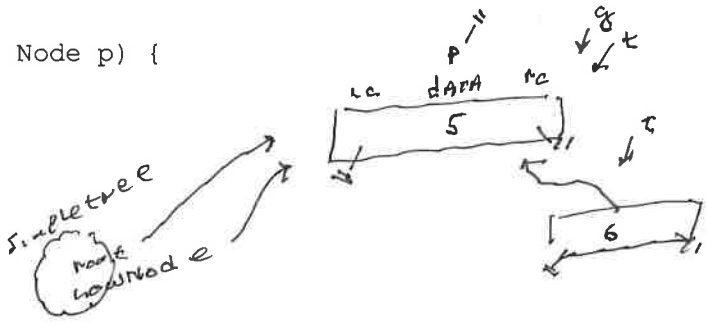
1.(a) Give the proper pre-condition at the point of code marked (1.a). head != null or LIST IS NOT empty (2 pts)
1.(b) Give the Big Theta of the code marked (1.b) (toString). $\Theta(N)$ (1 pts)
1.(c) Give the Big Theta of the code marked (1.c) (toString). $\Theta(N)$ (1 pts)
1.(d) Show the output of the code marked (1.d) 500 1 2 3 4 (4 Points)
1.(e) Show the output of the code marked (1.e) -1 2 3 4 (4 Points)
1.(f) Show the output of the code marked (1.f) 4 3 2 1 100 (4 Points)

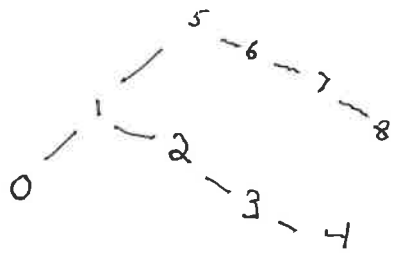
1.(g) Is it correct to say that the method foo() of the list class runs in $\Omega(2^n)$? Circle True or False (1 pt.)

1.(h) Is it correct to say that the method foo() of the list class runs in $\Omega(1)$? Circle True or False (1 pt.)

2. Study the execution of the following program. Five questions appear below. (10 points):

```
class Node {
    public int data;
    public Node lc;
    public Node rc;
    public Node p;
    public Node(Node lc, int x, Node rc, Node p) {
        this.lc = lc;
        this.data = x;
        this.rc = rc;
        this.p = p;
    }
}
public class SimpleTree {

    public Node root;
    public Node lowNode;

    public SimpleTree() {
        root = null;
        lowNode = null;
    }

    public void add(int x){

        if (root == null) {
            root = new Node(null,x,null,null);
            lowNode = root;
        }
        else {
            Node t = root;
            Node q = t;
            while(t != null) {
                if(x < t.data ) {
                    q = t;
                    t = t.lc;
                }
                else {
                    q = t;
                    t = t.rc;
                }
            } // end while
```





Key

```
            if(x < q.data) {
                q.lc = new Node(null, x, null, q);
                lowNode = q.lc;
            }
            else {
                q.rc = new Node(null, x, null, q);
                lowNode = q.rc;
            }
        }
    }
    public void traversal(Node r) {
        if(r == null) return;
        if(r.lc != null)traversal(r.lc);
        System.out.println(r.data);
        if(r.rc != null)traversal(r.rc);
    }
    public void traversal() {
        traversal(root);
    }

    public void traversal2() {
        Node q = lowNode;
        while( q != null) {
            System.out.println(q.data);
            q = q.p;
        }
    }
    public static void main(String[] args) {
        SimpleTree st = new SimpleTree();
        st.add(5);
        st.add(6);
        st.add(7);
        st.add(8);
        System.out.println("(2.a)");
        st.traversal2(); // show this output
        st.add(1);
        st.add(0);
        st.add(2);
        st.add(3);
        st.add(4);
        System.out.println("(2.b");
        st.traversal(); // show this output
        System.out.println("(2.c");
        st.traversal2(); // show this output

    }
}
```

2.(a) What will the program display at the traversal marked Question 2.a? (2 Points)

8 7 6 5

2.(b) What will the program display at the traversal marked Question 2.b? (2 Points)

0 1 2 3 4 5 6 7 8

2.(c) What will the program display at the traversal marked Question 2.c? (2 Points)

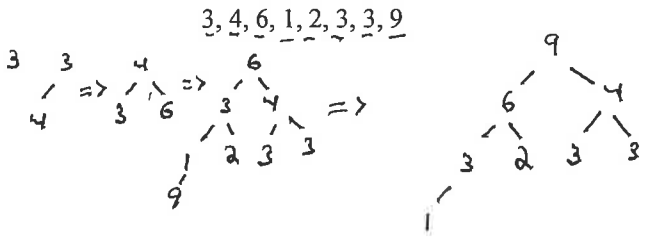
4 3 2 1 5

2.(d) Suppose we built a CLR complete tree and it was perfectly balanced (unlike the tree created above). Suppose too that the tree held n nodes in total. Provide a Big Theta value for traversal2(). $\Theta(\log N)$ (2 points)

2.(e) Suppose we built a CLR complete tree and it was perfectly balanced (unlike the tree created above). Suppose too that the tree held n nodes in total. Provide a Big Theta value for traversal(). $\Theta(N)$ (2 points)

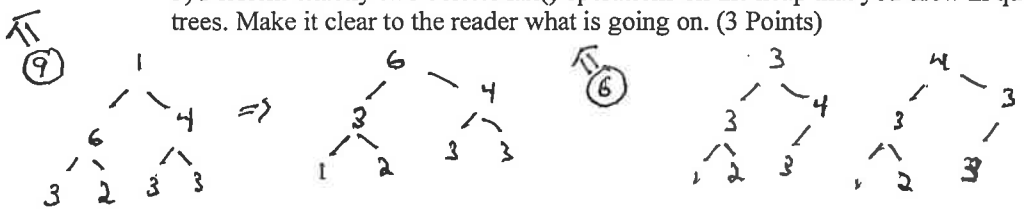
Heaps (12 points)

3) Insert the following 8 numbers into a max heap. Draw a new tree for each heap insertion. (4 Points)



4) What is the height of the tree that you drew in question 3? (A single node in a tree gives a height of 0.) (2 Points) 3

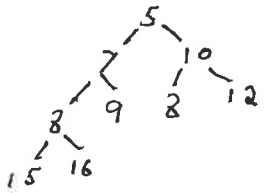
5) Perform exactly two deleteMax() operations on the heap that you drew in question 3. Draw the resulting trees. Make it clear to the reader what is going on. (3 Points)



6) Consider the following min heap implemented in an array. It is not quite correct. To make it a proper min heap exactly one swap must occur. What two numbers (child and parent) need to be swapped in order to make this a min heap? (3 points). PLACE CHECK MARKS NEXT TO THE TWO NUMBERS THAT NEED TO BE SWAPPED.

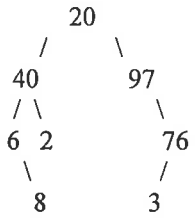
5
7
10 ✓
8
9
8 ✓
12
15
16

NOT correct



Binary Trees (16 points)

7. Parts (a), (b), (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)

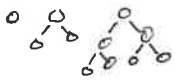
20, 40, 6, 8, 2, 97, 76, 3

(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, 8, 40, 2, 20, 97, 3, 76

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

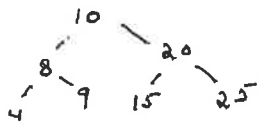
20, 40, 97, 6, 2, 76, 8, 3



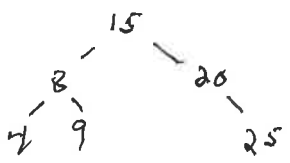
(d) In general, if a binary (at most two children per node) tree is perfectly balanced (unlike the tree pictured here) and complete with height h, how many nodes, in terms of h, will the tree have? (2 points) $2^{h+1} - 1$ Note, this tree has a perfectly flat bottom. We need the total number of nodes in terms of h. This is an exact answer, not Big O.

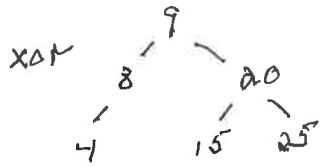
(e) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and complete with exactly k leaves. What is the height (in terms of k) of this tree? (2 points) $\log_2 K$ Note, this tree has a perfectly flat bottom. This is an exact answer, not Big O.

8. (a) Insert he following numbers into a Binary Search Tree. Draw the tree after all insertions are complete. (3 Points)
10, 8, 4, 9, 20, 15, 25



(b) Delete 10 from the final tree that you drew in 8 (a). Draw this final tree. (2 Points)
There are two possible answers. Either one is fine.





Key

Project Questions (20 points)

(9) Recall the Merkle-Hellman cryptosystem that we worked with in Project 1.

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 = {100, 9, 2, 105, 3, 7, 101} and **k** = 106. Is there a subset of X which sums to k?
Yes (Yes/No (1 point)

(b) The type of problem you were asked to solve in question 9 (a) is (Circle one answer): (1 Point)

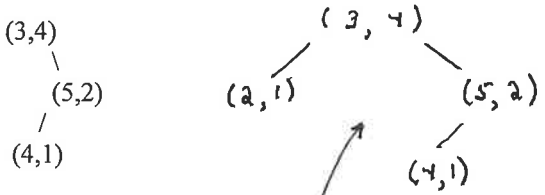
1. an optimization problem.
2. a problem that is impossible to solve.
3. a problem that has been proven to take exponential time to solve.
4. a problem that has been proven to take factorial time to solve.
5. a decision problem.

(c) Suppose Alice sends a message (K) to Bob. K is computed using Bob's Merkle-Hellman public key combined with the message M. The central idea behind Merkle-Hellman is that a potential eaves dropper could read the message M if the eaves dropper could (circle the one best option) (1 Points)

1. Find a subset of Bob's public key that sums to K.
2. Find K so that M is prime.
3. Modify Bob's public key.
4. Modify the super increasing sequence.
5. Find a subset of a super increasing sequence that sums to K.

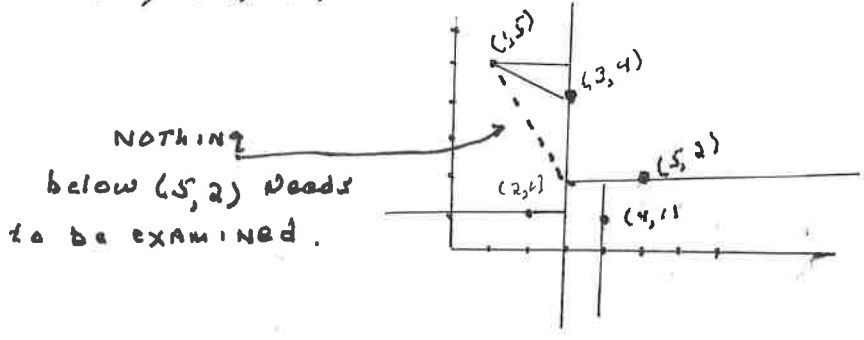
(d) Recall that a modular inverse of an integer b mod m is the integer b^{-1} such that $(b^* b^{-1}) \bmod m=1$. What is the modular inverse of 2 mod 11? 6 (2 Points)

(e) The following points, in a standard (x,y) coordinate plane, have been added to a 2-d tree. (3,4), (5,2), (4,1). The 2-d tree appears as follows:



Add the point (2,1) to this 2-d tree. Redraw the tree with this new point added. The first point, (3,4), breaks the plane vertically. (2 points)

(f) Consider the 2-d tree that you created, with the addition of (2,1), in (e). Suppose that we performed a nearest neighbor search for the point (1,5). Which points in the tree need to be examined? (3, 4), (2, 1), (5, 2) (2 Points)

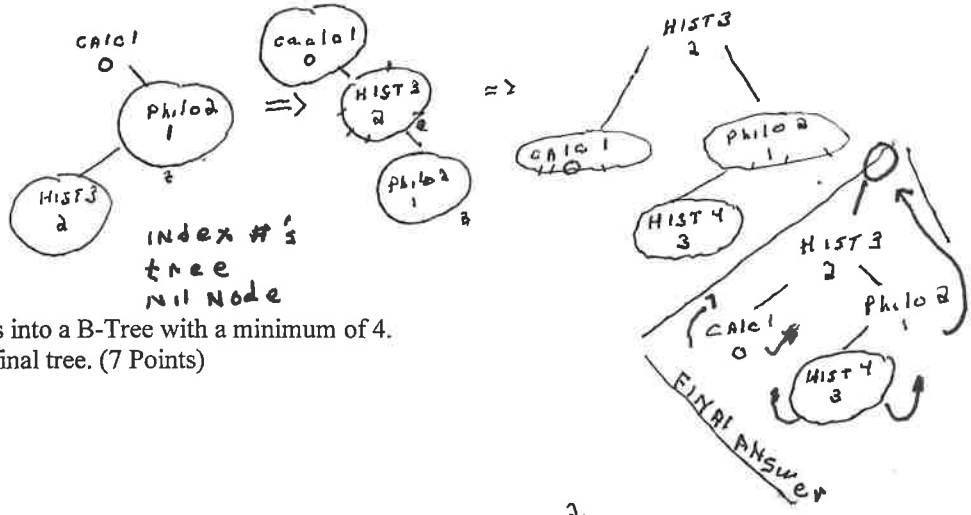


(g) In Project 3 we wrote a Red Black binary search tree. Suppose we are doing an insert of a course name into a Red Black Tree. Let T(n) be the number of operations required to do the insert. In the worst case, which of the following are true about T(n)? Circle all of those that are true. (You may or may not have more than one answer.) (4 Points)

1. T(n) ε O(Log(n))
2. T(n) ε O(1)
3. T(n) ε Ω(n^2)
4. T(n) ε O(n)
5. T(n) ε Θ(Logn)
6. T(n) ε O(2^n)
7. T(n) ε Ω(1)
8. T(n) ε Θ(n)
9. T(n) ε O(n!)

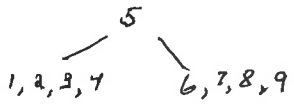
(h)The following is an execution of Project 3. Your task is to draw the Red Black Tree that exists after these commands are executed. Please be specific and **show all of the content of each node**. In addition, draw the nil node and show what nodes point to it. Color nodes red with a circle around the node. Black nodes have no circle. (7 Points)

Amy Calc1 Philo2 Hist3
Bill Calc1 Philo2 Hist4

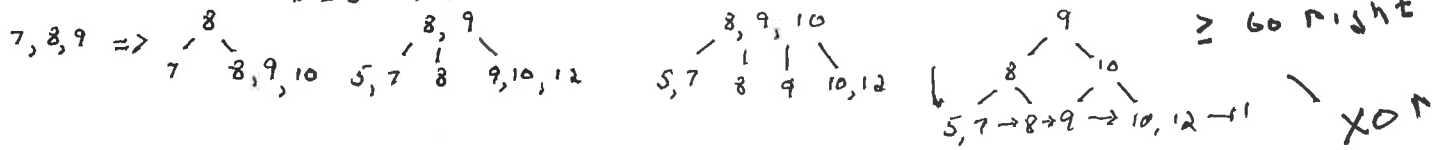


B Trees (21 points)

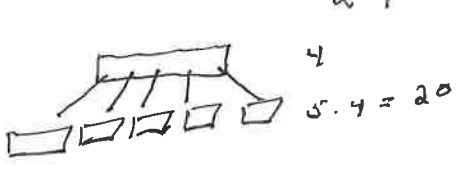
10. (a) Insert the following numbers into a B-Tree with a minimum of 4.
1,2,3,4,5,6,7,8,9 Draw the final tree. (7 Points)

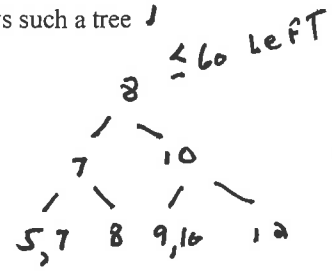


(b) Insert the following numbers into a B+ Tree with a minimum of 1.
9, 8, 7, 10, 5, 12. Draw each tree for partial credit. Draw the final tree. (7 Points)



(c) Consider a B-Tree with a minimum of 2. What is the exact maximum number of keys such a tree could hold if the tree were of height 1? 24 (7 Points)





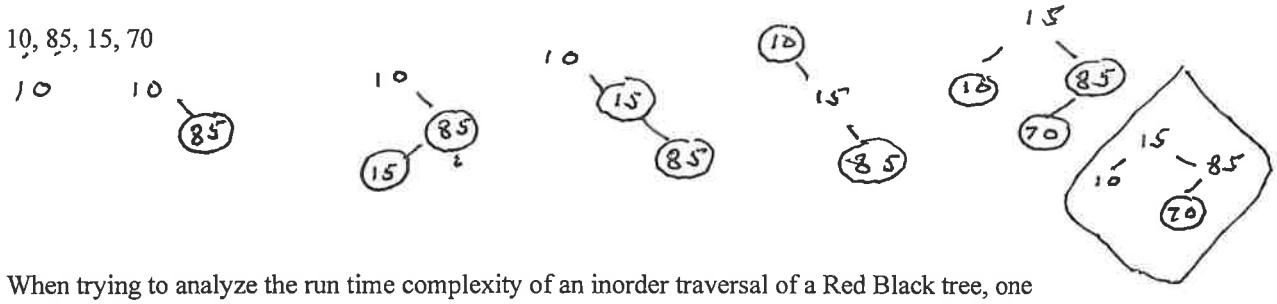
Key

Red Black Trees (8 points)

11. Red Black Trees

(a) Insert the following numbers, one by one, into a Red-Black Tree. Show the tree after each insertion. Draw RED nodes with a circle or a label 'R'. (5 points)

10, 85, 15, 70



(b) When trying to analyze the run time complexity of an inorder traversal of a Red Black tree, one should consider the worst case, average case and ~~best case~~ separately. Each case may have a different run time performance. Circle TRUE or (FALSE) (1 point)

(c) When trying to analyze the run time complexity of a Red Black tree insertion, one needs to consider the worst case, average case and best case. Circle TRUE or (FALSE) (1 point)

(d) What is the worst case runtime complexity of a Binary Search Tree insert operation? This is not a Red Black tree. Use Big Theta notation. (1 points) $\Theta(N)$

Graph Algorithms (26 points)

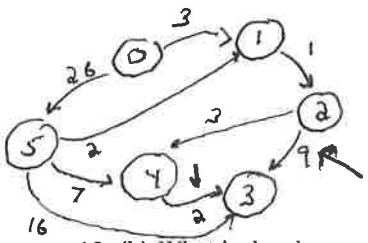
Consider the weighted, directed graph G_1. The graph is represented by an adjacency matrix m. If there is an edge from i to j with weight k then m[i,j] = k.

Matrix m

vertex	0	1	2	3	4	5
0		3				26
1			1			
2				9	3	
3						
4				2		
5		2		16	7	

G_1

12. (a) Draw the graph G_1 with circles and edges. (2 Points)



12. (b) What is the shortest path from the start node 0 to node 4 in the graph G_1? Your path must be a list of ordered pairs. (2 points) (0, 1), (1, 2), (2, 4)

Key #10

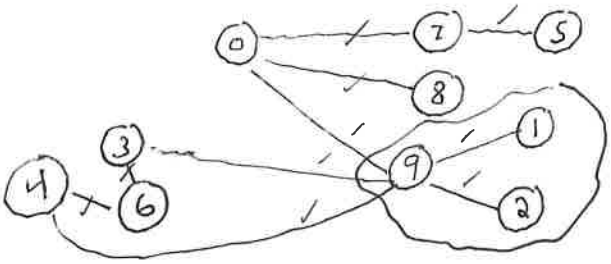
Consider the undirected graph G_2. The graph is represented by an adjacency matrix n. If vertex i shares an edge with vertex j then n[i,j] = T.

Matrix n

Vertex	0	1	2	3	4	5	6	7	8	9
0								T	T	T
1										T
2										T
3							T			T
4							T			T
5								T		
6				T	T					
7	T					T				
8	T									
9	T	T	T	T	T					

G_2

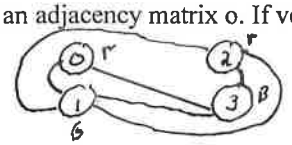
12. (c) Draw the G_2 graph with circles and edges. (2 points)



12. (d) Show the list of nodes that would be visited by a breadth first search in the graph G_2. We are starting from vertex 0. (3 points) 0, 7, 8, 9, 5, 1, 2, 3, 4, 6

ANY order ANY order LAST

Consider the undirected graph G_3. The graph is represented by an adjacency matrix o. If vertex i shares an edge with vertex j then o[i,j] = T.



Martrix o

vertex	0	1	2	3
0		T		T
1	T		T	T
2		T		T
3	T	T	T	

G_3

12. (e) What is the minimum number of colors that we could color G_3 with? (3 Points) 3

Key 11

Consider the undirected graph G_4. The graph is represented by an adjacency matrix p. If vertex i shares an edge with vertex j then p[i,j] = T.

Martrix p

vertex	0	1	2	3
0		T	T	T
1	T		T	T
2	T	T		T
3	T	T	T	

G_4

12. (f) What is the minimum number of colors that we could color G_4 with ? (3 Points) 4

Consider the undirected graph G_5. The graph is represented by an adjacency matrix q. If vertex i shares an edge with vertex j then q[i,j] = T.

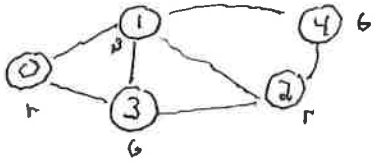
Martrix q

fixed ow bound

Vertex	0	1	2	3	4
0		T		T	
1	T		T	T	T
2		T		T	T
3	T	T	T		
4		T	T		

G_5

12. (g) What is the minimum number of colors that we could color G_5 with ? (3 Points) 3



12. (h) Assume that you have routines to mark a vertex, process a vertex and place a vertex in a queue. You also have a routine that allows you to access each unmarked neighbor of a vertex. Write an algorithm that determines if a path exists starting at vertex v and ending at vertex w. Your algorithm will use the queue to perform a breadth first traversal on the graph. You need not code this in Java. But the algorithm should be clear to the reader. (8 points)

The signature of the method isPath looks like the following:

```
// returns true if and only if there is a path from vertex v to vertex w.
boolean isPath( int v, int w);
```

1) using a queue
2) handling unmarked neighbors

```
g.add(v)
while !g.isEmpty() {
    u = g.delete()
    mark u
    if u == w return true
    For each unmarked neighbor N of u {
        g.add(N)
    }
}
return false
```