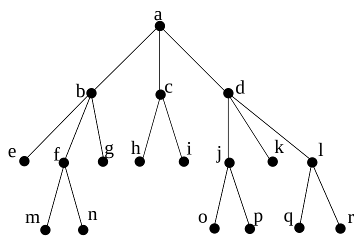
Math 210
Exam 4 **Name:**__________________________________

Instructions: You will have 55 minutes to complete this exam. The credit given on each problem will be proportional to the amount of correct work shown. Answers without supporting work will receive little credit.

1. (3 points each) Answer the following questions based on the rooted tree shown below:



(a) List the children of vertex d.

j, k, l

(b) List the ancestors of vertex n

a, b, f

(c) Find the number of leaves in this rooted tree.

This tree has 11 leaves.

(d) List all level 2 vertices in this rooted tree.

The level 2 vertices are: e, f, g, h, i, j, k, and l.

(e) Find the height of this rooted tree.

This rooted tree has height 3.

(f) How many vertices would you need to add to this tree in order to make it a *full* 3-ary tree?

To make this a full 3-ary tree, we would need to add 4 vertices (we add an additional child to c, f, j, and l).

(g) How many vertices would you need to add to this tree in order to make it a *complete* 3-ary tree?

To make this tree complete, we need every leaf to be at level 3. To accomplish this, we add the 4 vertices mentioned above, and then we extend the tree to level 3 at each branch by adding three children to each of the level two leaves, requiring 18 additional vertices. This means that we must add a total of 22 vertices to make this a complete tree (well, I suppose you could add many more vertices to make it complete of height 4 or even more, but none of you attempted to do this – you sensibly tried to create the smallest complete tree that has our original tree as a subtree)

2. (a) (5 points) Find the order that you would visit the vertices of this tree if you use preorder traversal to visit the vertices.

a, b, e, f, m, n, g, c, h, i, d, j, o, p, k, l, q, r

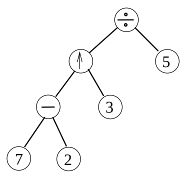
(b) (5 points) Find the order that you would visit the vertices of this tree if you use inorder traversal to visit the vertices.

e, b, m, f, n, g, a, h, c, i, o, j, p, d, k, q, l, r

(c) (5 points) Find the order that you would visit the vertices of this tree if you use postorder traversal to visit the vertices.

e, m, n, f, g, b, h, i, c, o, p, j, k, q, r, l, d, a

3. (7 points) Draw the ordered rooted tree corresponding to the postfix expression: $72 - 3 \uparrow 5 \div$, then compute the value of the expression.



Notice that traversing the computational tree above using postorder recovers the original expression. Carrying out this computation, we find $7 - 2 = 5$, $5^3 = 125$, and $\dfrac{125}{5} = 25$.

4. (12 points) Use a proof by induction to prove that a tree with n vertices has $n-1$ edges.

Base Case: Consider a tree with $n = 1$ vertices. Notice that such a tree must have no edges, since a tree is a simple graph and hence has no loop edges. Then the theorem is true when $n = 1$.

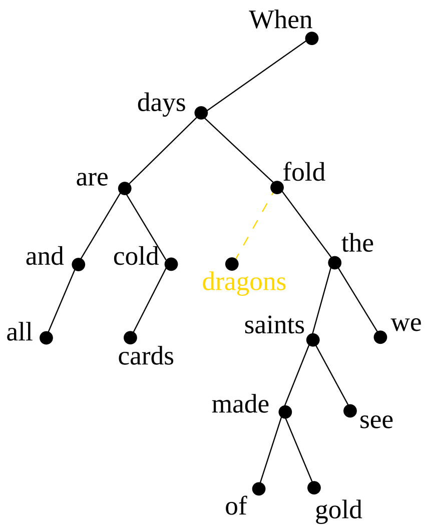
Inductive Step: Suppose that any tree with k vertices has $k-1$ edges.

Consider a tree T with $k+1$ vertices. Note that since $k \geq 1$ that $k+1 \geq 2$, and hence T has at least one edge. We claim that there must be at least one edge in T that is a pendant edge.

To see this, consider a maximal simple path in T. Since a tree has no circuits and the graph T is finite, the final edge in this path must be a pendant edge. Let v be the final vertex in the degree sequence of this path. Let u be the unique vertex adjacent to v. If we remove remove vertex v along with the pendant edge $\{u, v\}$, we obtain a graph T'. We claim that T' is a tree. Notice that since the only edge we removed was a pendant edge, and we also removed the vertex v, then T' is connected. Also note that since edges were removed and no edges new were added, there are no simple circuits in T'.

Then T' is a tree with k vertices. Hence, applying the induction hypothesis to T', this tree must have $k-1$ edges. But then the original tree T has k edges. This proves the theorem. □.

5. (a) (8 points) Draw a binary search tree for the phrase: "When days are cold and cards all fold the saints we see are made of gold."



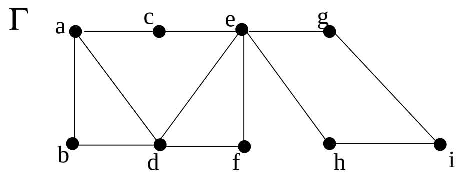
(b) (3 points) How many comparisons are needed to find the word "gold" in this tree?

Using the tree constructed above, we must perform 7 comparisons in order to find "gold" in the tree. Note that we need to start at the root vertex and compare every vertex along the path from the root to the node labelled "gold", including the final node.

(c) (3 points) How many comparisons are needed to add the word "dragons" to this tree?

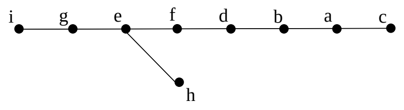
Similar to the previous question, we begin at the root and perform comparisons until we are able to add a new node labelled by "dragons". We must perform 3 comparisons to do this (see the new node placed in the figure above).

6. (6 points each) Given the following graph:



(a) Use a depth first search starting at the vertex i to find a spanning tree for this graph.

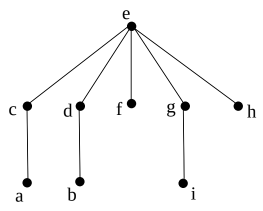
There are several correct solutions to this problem. Here is one of them:



Note that you must have extended all paths as far as possible in your construction, and you must have used "backtracking" appropriately to find the proper node to add a new branch to the tree.

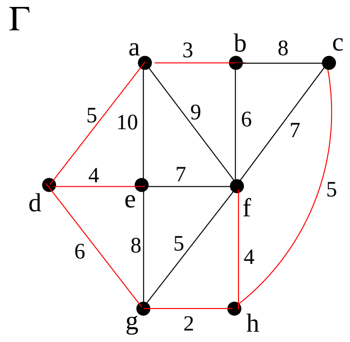
(b) Use a breadth first search starting at vertex e to find a spanning tree for this graph. Use the standard alphabetic ordering to order the vertices added during each step.

There are several correct solutions to this problem. Here is one of them:



Note that you were asked to start at vertex e and you were asked to put children in alphabetical order. Not doing so impacts the construction of level 2 in your tree.

7. (8 points) Given the following weighted graph, use Prim's Algorithm to find a minimum spanning tree for the graph. Be sure to indicate the order that the edges were added to the spanning tree.



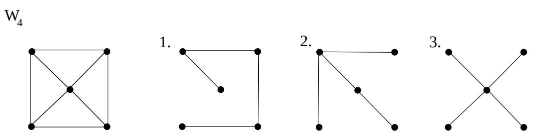
The following tables shows the order in which edges are added to a spanning tree using Prim's Algorithm. Note there are two equivalent choices in step 4, so there were two possible correct answers to this question. The spanning tree that results from the first result is shown above.

Step:	Edge:	Weight:
1.	$\{g, h\}$	2
2.	$\{f, h\}$	4
3.	$\{c, h\}$	5
4.	$\{d, g\}$	6
5.	$\{d, e\}$	4
6.	$\{d, a\}$	5
7.	$\{a, b\}$	3

Step:	Edge:	Weight:
1.	$\{g, h\}$	2
2.	$\{f, h\}$	4
3.	$\{c, h\}$	5
4.	$\{b, f\}$	6
5.	$\{a, b\}$	3
6.	$\{a, d\}$	5
7.	$\{d, e\}$	4

The total weight if the spanning tree(s) formed by this algorithm is: $2 + 4 + 5 + 6 + 4 + 5 + 3 = 29$.

8. (6 points) Find the number of nonisomorphic spanning trees that W_4 has.



There are three non-isomorphic spanning trees of W_4. They can be seen in the figure above.

9. (10 points) Suppose that you are given 5 envelopes and a balance scale. Each envelope contains an identical money order. The money orders are for \$1, \$10, \$50, \$200, and \$600 respectively. The envelopes have also been weighted so that the weight of the envelopes correlates to the dollar amount inside (the heavier the envelope, the more money). You will be allowed to choose one envelope and open it. You may also use the scale to compare envelopes. However, for each time you use the scale, you lose half of the dollar amount on the money order from the envelope you choose to open. Describe the strategy you would use to maximize the amount of money you receive. You should also state the amount of money you expect to receive.

Clarification: The description of this problem states that the weight of an envelope *correlates* with the amount on the money order inside (the heavier the envelope, the more money). This tells us that weighing the envelopes will allow us to determine the relative value of the envelopes. However, many of you took this to mean that the weights of the envelopes are *proportional* to the dollar amount on the money order inside. This was an unfounded assumption and went well beyond what was stated in the problem description. We know that heavier envelopes have more valuable money orders inside, but we have no idea how much each weight differs from the others. They may only differ by a very slight amount. Those that made this assumption greatly simplified the problem and arrived at strategies that would not work for the actual situation that was described. After re-reading the problem description and conferring with a few colleagues, I decided that the fault was on the side of those who made an incorrect interpretation, not on the way the question was phrased, so I assessed a significant penalty to those who used a strategy based on this assumption.

That being said, how can we solve this problem given the notion that heavier envelopes have more valuable money orders, but the difference in their weights may be very slight? First, one should conclude that weighing more than one envelope per side together will not yield useful information, since pairing a low value envelope with a high value envelope may not have more total weight that two medium value envelopes. For this reason, effective strategies should be based on weighing a single envelope against another single envelope.

A strategy for identifying the most valuable envelope in the minimum number of weightings would be as follows: We label the envelopes "a", "b", "c", "d", and "e".

Weigh "a" vs "b". Keep the heavier of the two – call this w_1. Then weigh "c" vs "d". Keep the heavier of the two – call this w_2. Weigh w_1 vs "e", keep the heavier of the two – call this w_3. Finally, weigh w_2 vs w_3. The heavier of these two must be the \$600 money order.

Note that we did four total weighings to identify the most valuable envelope. If we open it, we would take home: $\frac{600}{2\cdot2\cdot2\cdot2} = \$37.50$.

Even if we are convinced that it takes 4 weighings to find the most valuable envelope, we are still left with the question, would it be better to stop at an earlier point and open an envelope while they are all have more value left. For example, if we were sure that both the \$200 and the \$600 were the only two left after 3 weighings and we opened one of them, we would get $\frac{600}{2^3} = \$75$ half the time and $\frac{200}{2^3} = \$25$ half the time. That gives an average expected return of \$50, which in some sense is better than our solution above. However, this outcome only happens if the \$600 envelope and the \$200 envelope are not compared to each other in any of the 3 previous weighings.

Notice that the \$600 envelope and the \$200 envelope will only both survive the weighing process if one of them is given the label "a", "b", or "e" and the other is given the label "c" or "d". Otherwise, they will be weighed against each other before the last pairing, so we would be left with the \$600 envelope and a lower value envelope (either \$50 or \$10). The nightmare scenario is weighing \$600 vs \$200 first, then \$10 vs \$1 second, and weighing \$600 vs \$50 for the third weighing, leaving \$600 and \$10 yielding either \$75 or \$1.25. This still gives an average expected return of \$38.125, which in some sense is better than our solution above. Based on this, we may want to take one step back and think about the options available to us after only two weighings, which I will not do here in the interests of time. The one weighing case is straightforward: if we pick an envelope at random at the beginning, we have a 1 in 5 chance of getting any particular envelope, meaning that our average expected return is: $\frac{600+200+50+10+1}{5} = \frac{861}{5} = \$172.20$.

For this reason, the answer to this question depends on how you define "maximize". If maximize means the largest guaranteed payout, then the original solution outlined above gives the best outcome. If maximize is defined probabilistically, then we would want to do less than 4 weighings and pick a random envelope, which would given a larger average payout. To decide whether to stop after 0, 1, or 2 weighings would require a more careful analysis.