Question 1. (12 marks). *General.* [illegible]

Answer the following questions by providing a **very brief and direct** answer.

1. What is the difference between primary clustering and secondary clustering?

2. Given a hash table of size H, where H is a prime number. What is the minimum number of probes necessary before the table is declared full?

3. A 2-tree is a binary in which each node has exactly two empty subtrees or two non-empty subtrees. True or false: any 2-tree has an odd number of nodes?

4. Define the spanning tree of a graph?

5. Why is the following solution to the Towers of Hanoi problem incorrect?

```
/* move n disks from src to dest */
void move (int n, char src, char dest, char temp)
{
 if (n == 1) {
   cout << "move top disk from " << src << " to " << dest << endl;
 }
 else {
   /* move top disk from src to temp */
   cout << "move top disk from " << src << " to " << temp << endl;

   /* move (recursively) n-1 disks from src to dest */
   move (n-1,src,dest,temp);

   /* move disk from temp to dest */
   cout << "move top disk from " << temp << " to " << dest << endl;
 }
}
```

6. We decide to increase the size of a hash table when the density factor λ becomes greater the 0.5. In order to do so, we must rehash all the elements already in the table. Explain why this rehashing is necessary.

7. Show that $f(n) = n!$ is $O(n^n)$.

8. All graphs are R-colorable for R>4, but not necessarily using the degree<R rule. True or False?

9. Any graph with n vertices has at most n(n-1)/2 edges. True or False?

10. A greedy algorithm is always optimal. True or False?

11. What is the minimum number of nodes in a heap of height h?

12. Does Dijkstra's shortest-path algorithm work if there are edges with negative weights in the graph?

Question 2. (4 marks). *Complexity Analysis.*

The following code is excerpted from a program that computes a special function of the elements of two arrays a and b. The function weird runs in O(1). What is the complexity of the excerpt of code? Show the details of your analysis.

```
for (i=0 ; i < n ; ++i) {
   c[i] = 0;
   for (j=i ; j < n-i ; ++j) {
        c[i] = weird(a[i][j], b[i][j]);
   }
}
```

Question 3. (5 marks). *Stacks.*

The data words below are available in left-to-right order for insertion in a stack:

"it" "organized" "a" "program" "as" "is" "mind" "as" "the" "produced" "that"

The operation **S** inserts (pushes) the leftmost word (not yet accessed) into the stack; the operation **X** removes (pops) the topmost word from the stack.

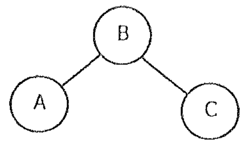
Give a sequence of operations that will result in the data being removed from the stack in the following left-to-right order:

"a" "program" "is" "as" "organized" "as" "the" "mind" "that" "produced" "it"

Write the sequence in the spaces provided below.

Question 4. (8 marks). *Trees.*

(a) (4 marks). In the *reverse inorder* traversal of a tree, the right subtree of each node is first traversed (recursively in reverse inorder), the node then is visited, and finally the left subtree of the node is traversed (recursively in reverse inorder). That is, the order of the traversal is RNL. For example, the reverse inorder traversal of the tree shown below is C B A.



Given the following *reverse inorder* and preorder traversals of *a* binary tree, draw *the* tree.

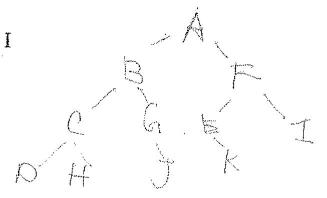
R N N L

Reverse Inorder: I F K E A G J B D C H

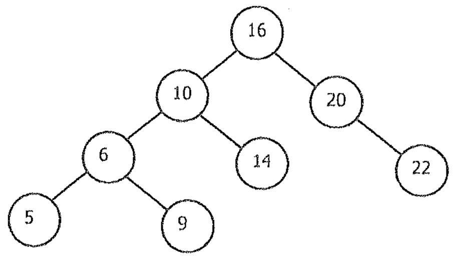
N L R

Preorder: A B C H D G J F E K I

N



(b) (4 marks). The following is an AVL tree. Show where a node with the key "4" would be inserted onto the tree. If the insertion unbalances the tree, then re-balance it, and redraw the tree after re-balancing it.



Question 5. (8 marks). *Heaps.*

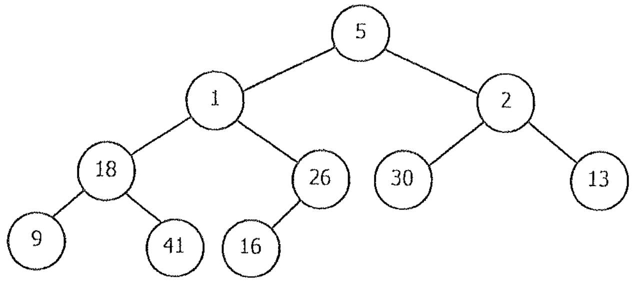
Write a procedure that will delete the element with index i from the heap, and restore heap properties to the resulting, smaller list. You may assume that the heap procedures described in class or in your textbook (i.e., bubble_up, bubble_down, build_heap, etc) are available to you (i.e, you do not have to implement them; simply use them if you wish).

The complexity of your procedure should be no worse than O(log n), where n is the size of the heap array.

Your code should not exceed a few lines. Clarity is essential. Excessively long and/or solutions unclear will be penalized!

Question 6. (9 marks). *Heaps.*

The following is a binary tree.



(a) (3 mark). Draw the binary tree using its array representation.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

(b) (3 marks). Build a heap from the elements of the array. Show only the array representation of the heap after the build is done.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

(c) (3 marks). Start with the heap in part **(b)** and insert a new element with the key "15" and restore the heap if necessary. Show only the array representation of the heap after the heap is restored.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Question 7. (12 marks). *Hash Tables.*

Assume a hash table with size H = 13. Double hashing is used to resolve collection. If double hashing fails to resolve a collision, then linear probing is used. The following are the two hash functions:

h_1: index = (key mod T) + 1
h_2: index = (key mod (T-2)) + 1

Show the contents of the hash table after the following operations have been performed. Indicate next to each operation the number of *probes* performed.

1. insert 7
2. insert 21
3. insert 33
4. insert 150
5. insert 9
6. insert 36

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	

Question 8. (10 marks). *Game Trees.*

Consider the popular game of tic-tac-toe. Assume there is a function *evaluate* that accepts a board position and returns a numerical value that represents how "good" the positions seems to be for the machine (the larger the value, the better the position). The value returned by the function is the number of rows, columns and diagonals remaining open for the machine (X) minus the number of rows, columns and diagonals remaining open for the human (O). The value returned is 9 for a position that wins and –9 for a position that loses. The following are examples of board positions and their respective evaluate function values.

X	O	X
	X	
		O

3-1=2

X	O	
X	X	
		O

2-1=1

X	O	
O	X	
		X

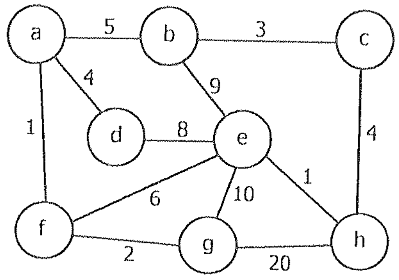
9

Given the board position shown below (it is the machine's turn to make a move, i.e., place an X), show the 2-ply (i.e., 2 more moves) game tree generated the machine. Using the min-max algorithm, indicate which move the machine should take.

X	O	
	X	
		O

Question 9. (10 marks). *Graphs.*

Shown below is an undirected weighted graph G.



(a) (6 marks). Give the breadth-first and depth-first traversals of G. Start at vertex **a**. In the case of the depth-first traversal, use the stack-based algorithm described in class.

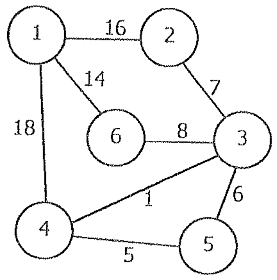
Breadth-first traversal:

Depth-first traversal:

(b) (4 marks). Show a minimum cost spanning tree (MCST) rooted at vertex **a** in the above graph. Indicate your answer on the graph by highlighting the vertices and edges that belong to MCST.

Question 10. (9 marks). *Graphs.*

Use Dijkstra's shortest-path algorithm to determine the cost of the shortest path between and vertex 1 and vertex 3. Show the progress of the algorithm by showing, for each iteration of the algorithm, the contents of the distance array w (called D in your textbook) that contains the estimate of the costs of the shortest paths for all vertices.



1	2	3	4	5	6

1	2	3	4	5	6

1	2	3	4	5	6

1	2	3	4	5	6

Cost of SP:

1	2	3	4	5	6

1	2	3	4	5	6

Question 11. (8 marks). *Grammars.*

(a) (4 marks). Given a grammar G=(T,N,S,P), where T={a,b,c}, N={Z,A} and S=Z. The following are the productions P:

$Z \rightarrow A$

$A \rightarrow a\ A\ b\ A$

$A \rightarrow a\ A$

$A \rightarrow c$

Show the parse tree for the sentence **a a c b c**. Is this grammar unambiguous? Justify your answer!!

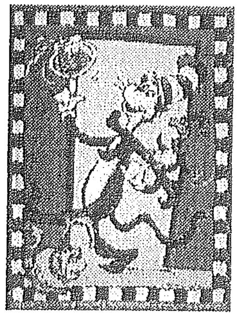
(b) (4 marks). Write a grammar for the following if-then-else construct in a programming language:

```
if (condition) then
   stmt
else                              if (condition) then
   stmt               or             stmt
endif                             endif
```

where "condition" may take only the form of "variable == variable" or "variable != variable". A "variable" is a single lower-case character. A "stmt" may take only the form "variable = variable + variable" or "variable – variable". The "else" part of the if-then-else construct is optional.

Question 12. (5 marks). *Recursion.*

(An homage to Theodore Seuss Geisel) [Owen Astrachan, Duke University]



The Cat in the Hat is a nasty creature,
But the striped hat he is wearing has a rather nifty feature.

With one flick of his wrist he pops his top off.

Do you know what's inside that Cat's hat?
A bunch of small cats, each with its own striped hat.

Each little cat does the same as line three,
All except the littlest ones, who just say "Why me?"

Because the littlest cats have to clean all the grime,
And they're tired of doing it time after time!

The cat walks into a messy room that he needs to clean. Instead of doing the work, he decides to have the smaller cats inside his hat do the work (call them helper cats). Each helper cat also has smaller helper cats in his own hat, and so on. Eventually, the cats reach a smallest size. These smallest cats have no additional cats in their hats. These unfortunate smallest ones have to do the cleaning:

The smallest cats are of height one;
these are the cats that get the work done.

The number of cats inside each (non-smallest) cat's hat is some unknown constant, *d*. The height of these cats-in-a-hat is *1/(d+1)* times the height of the cat whose hat they are in. Given *N*, the height of the initial cat and *H*, the number of helper cats (i.e., cats of height one), find the number of cats that are not doing any work (i.e., cats of height greater than one).

p15 blank