CSC 190S 1998 Final

[Que]stion 1 (10 marks). *General.*

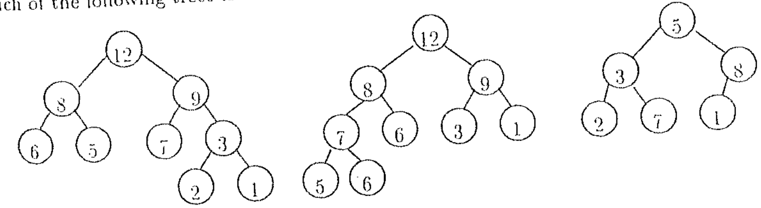
[Pro]vide *brief* answers to the following. Unless otherwise indicated, each part is worth 1 mark.

1. What is the minimum and maximum number of nodes in a heap of height h.

2. In what step of the mergesort algorithm are the elements actually sorted (swapped)?

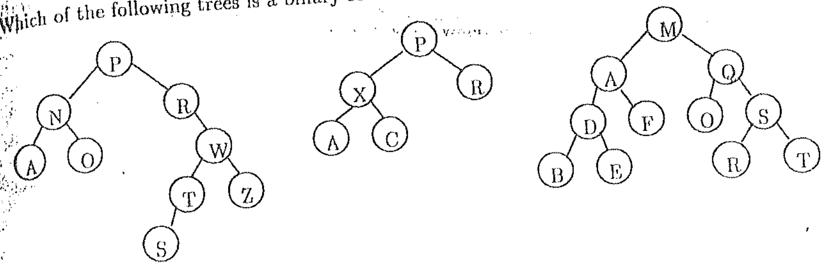
3. How many edges ($|E|$) are there in a completely connected graph of $|V|$ vertices?

4. Which of the following trees is a max-heap?



5. In one sentence, explain how the Heapsort program discussed in class can be modified to sort the array in decreasing order instead of increasing order.

Which of the following trees is a binary search tree?



7. What, if any, is the advantage of Heapsort over Quicksort?

8. (2 marks) We wish to write the elements of a binary search tree to a disk file, with no indication of the tree's structure. If we wish to then read in the file and restore the tree to its original structure using `BST::insert()`, what tree traversal should we use when writing the disk file in the first place?

9. What advantage does *double hashing* have over *linear* and *quadratic probing*?

on 2 (12 marks). ***Recursion***

ibonacci sequence is as follows: 1, 1, 2, 3, 5, 8, ... where each succesive term is equal to of the two previous terms. The first number is defined to be F_1, the second F_2, and so that $F_6 = 8$ for example.

marks). Assume you are implementing the Fibonacci function as a recursive function. /are the general case(s)? The terminal/base case(s)?

(4 marks). Implement a RECURSIVE Fibonacci function with the following prototype. parameter passed is invalid, your function should return 0.

```
Fibonacci(int n)
```

(… marks). Now implement a version of `Fibonacci()` that is iterative (does not use …on). Your solution must work for arbitrarily large n.

```
Fibonacci(int n)
```

…) (1 mark). The recursive version of `Fibonacci()` is very inefficient, and not just due to the overhead of recursion. Explain why.

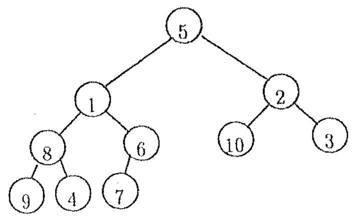
…on 3 (8 marks). *Quicksort*

…marks). Describe what happens in the worst-case scenario for quicksort when the pivot … taken from the middle of the array.

… (… marks). An alternative method for choosing the pivot value in quicksort is to take the … of the first, middle and last elements of the unsorted part of the array. Compare the … complexity of the worst-case scenario for this method of choosing the pivot value with … considered above.

...on 4 (12 marks). *Heaps.*

...llowing is a complete binary tree.



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

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

b. (3 marks). Build a max-heap from the elements of the array. Show **both** the array and ... representation of the heap after the build is done.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

... (3 marks). Start with the heap in part b) and delete the element with the largest key ... restore the heap as needed. Show **only** the tree representation of the heap after the heap is restored.

...tion 5 (13 marks). *Hash Tables.*

...me a hash table with size $T = 11$, and the following hash function:

...dex = ($key \bmod T$).

...(3 marks). Show the contents of the hash table after the ...wing operations have been performed. Indicate next to each ...ation the number of *probes* performed. Assume that linear ...hing is used to resolve collisions and that the hash table is ...tially empty.

1. insert 7.
2. insert 19.
3. insert 22.
4. insert 40.
5. insert 51.
6. insert 63.

index	key
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

b) (3 marks). Redo part a., but this time resolve collisions using quadratic probing where the probe step is multiplied by a second hashing function, *i.e.* the probe sequence is given by $h(\text{key}) + i^2 h_2(\text{key}) \bmod T$ where $h_2(\text{key}) = 7 - (\text{key} \bmod 7)$. Again, indicate next to each operation the number of *probes* performed and assume that the hash table is initially empty.

1. insert 7.
2. insert 19.
3. insert 22.
4. insert 40.
5. insert 51.
6. insert 63.

index	key
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

...ks). Calculate the load factor α for the hash tables in parts a) and b). What is the ...verage cost for both successful and unsuccessful searches in part a)?

...marks). What is the purpose of a *tombstone*? Give an example using <u>either</u> part a) or part b).

...(2 marks). Remove the element with value 7 from the hash table in part a), and re-hash the table. (Re-insert elements into table in the order which they appeared in the table after the deletion, and use linear probing.)

index	key
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Question 6 (8 marks). *Binary Search Trees*

a) ([illegible] marks) Write an <u>inorder</u> tree traversal which deletes all the nodes in a BST.

b) (5 marks) You have a binary search tree class named BST. Write an iterative (non-recursive) member function which looks for the first instance of a particular value stored in a BST. Return NULL if the value is not found in the tree. Assume the root of the tree is stored in a member variable `BinNode *root`. Use the following prototype:

```
BinNode *BST::find(const BELEM &val){



}
```


Question 7 (14 marks).

A binary tree can be used in coding problems such as in encoding and decoding messages transmitted in Morse code, a scheme in which characters are represented as sequences of dots (.) and dashes (-), as shown in the following table

[illegible]	E .	I ..	M - -	Q - -.-	U ..-	Y -.- -	2 ..- - -	6 -....
[illegible]	F ..-.	J .- - -	N -.	R .-.	V ...-	Z - -..	3 ...- -	7 - -...
C -.-.	G - -.	K -.-	O - - -	S ...	W .- -	0 - - - - -	4 ....-	8 - - -..
[illegible]	H	L .-..	P .- -.	T -	X -..-	1 .- - - -	5	9 - - - -.

In this case, the nodes in a binary tree are used to represent the characters, and the links from a node to its children are labeled with a dot or dash, according to whether they lead to a left child or to a right child, respectively. The root of the tree is an empty node. In the class declaration given below, the pointers `left` and `right` are set to `NULL` if the corresponding subtree doesn't exist.

```
class MorseCodeTree {
public:
  struct Node {
    char letter;
    Node *left, *right;
  };

  typedef Node *NodePtr;

  NodePtr root;
};
```

a) (4 marks). Draw a representation of the first 4 levels of the tree (*i.e.*, the 4th level has 8 nodes).

arks) Complete the following function:

```
eCode' is a null-terminated string containing the characters
r '-' representing the Morse code of a character and 'tree'
a MorseCode tree.  Return the character represented by 'MorseCode'.
etCharacter( const MorseCodeTree& tree, const char MorseCode[ 6 ] )

}
```

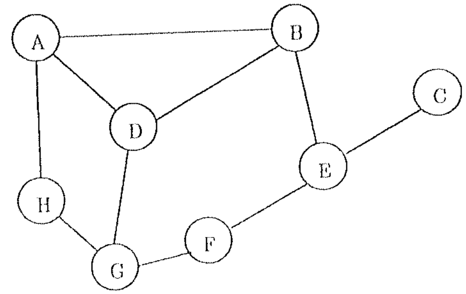
(6 marks) Complete the following function:

```
// 'ch' is a character whose Morse code is to be returned as a null-
// terminated string of '.' and '-' in the array 'MorseCode'. 'tree'
// is a Morse code tree.
void getMorseCode( const MorseCodeTree& tree, char ch, char MorseCode[ 6 ] )
{

}
```


tion 8 (13 marks). *Graphs.*

below is an undirected 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:

Depth-first:

b. (7 marks). Write a function to determine if a graph G contains any cycles. Use the ADT from the text. [Hint: This can be based on a modified DFS traversal of the graph.]

tion 9 (4 marks). *Traveling Salesman*

mark). Explain what is meant by a *simple cycle* in a graph G.

b) (3 marks). The **Traveling Salesman Problem** is defined as follows: Given a complete, undirected graph G with distances assigned to each edge in the graph, find the shortest simple cycle which includes every vertex.

An algorithm is in NP if a "guess" at the correct solution can be verified in polynomial time. The brute-force solution to TSP is exponential. Is TSP in NP? Explain your answer.

n 10 (6 marks). General Trees

array representation of a complete binary tree, the following equations allow finding x of various nodes which are related to the node stored in array index r:

$$\text{Parent}(r) = (r-1)/2 \text{ if } 0 < r < n$$
$$\text{LeftChild}(r) = 2r+1 \text{ if } 2r+1 < n$$
$$\text{RightChild}(r) = 2r+2 \text{ if } 2r+2 < n$$
$$\text{LeftSibling}(r) = r-1 \text{ if } r \text{ is even and } 0 < r < n$$
$$\text{RightSibling}(r) = r+1 \text{ if } r \text{ is odd and } r+1 < n$$

n is the number of elements stored in the tree. Derive similar equations for an array-trinary tree which allow one to find the parent, left sibling, right sibling, left child, child and right child of a node whose index is r. [Hint: you may find it helpful to draw t similar to Figure 5.11 to assist your thinking]