Data Structures: Practice Midterm Solutions

1. **Mathematical Background**

   a. f(N) is O(g(N)) if there are positive constants c and N_0 such that $\underline{f(N) \le cg(N)}$ $\underline{\text{for } N \ge N_0}$

   b. Show that 373N+100 is O(N) (by selecting appropriate constants c and N_0). Here, f(N) = 373N+100 and g(N) = N. There are many different solutions. One solution: Select c = 374 and N_0 = 100. Then,
   $cg(N) = 374N = 373N + N \ge 373N+100$ for $N \ge 100$

   c. If T(N) is the run time of the following function, the following statements are true (the other two are false):
      (i) T(N) is $O(2^N)$
      (ii) T(N) is $\Omega(\log N)$

```
int FunWith(N){
if (N == 0) return 1;    /* 1 */
else return FunWith(N-1) + FunWith(N-1);} /* 2 */
```

   Here's why. Line 1 takes a constant amount of time c_0 (for N = 0) and the "if…else" and "+" in line 2 takes constant time c, plus the time for the two recursive calls. Therefore, the recurrence relation for T(N) is:

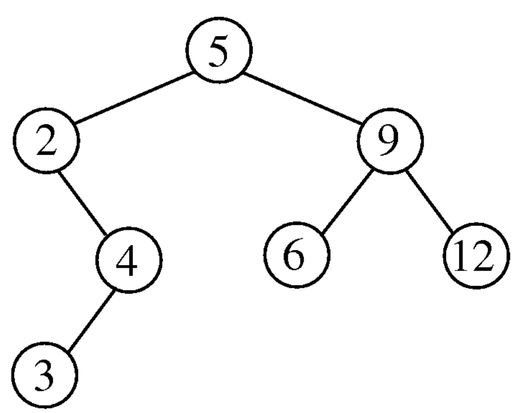
$$T(N) = 2T(N-1) + c$$
$$= 2(2T(N-2) + c) + c = 2(2(2T(N-3) + c) + c) + c = \ldots$$
$$= 2^N T(N-N) + c(2^{N-1}+\ldots+2^1 + 2^0) = 2^N c_0 + c(2^N-1) = \Theta(2^N)$$

   This is both $O(2^N)$ and $\Omega(\log N)$ but not $\Theta(N)$ or $o(2^N)$.

2. **Trees and Stacks**

   a. Draw the final tree that results from inserting the integers 5, 2, 4, 3, 9, 12, 6 (in that order) into an empty binary search tree with no balance conditions.

   Solution:



b. What is the sequence of elements that results from a preorder traversal of your tree in part (a)?
5 2 4 3 9 6 12

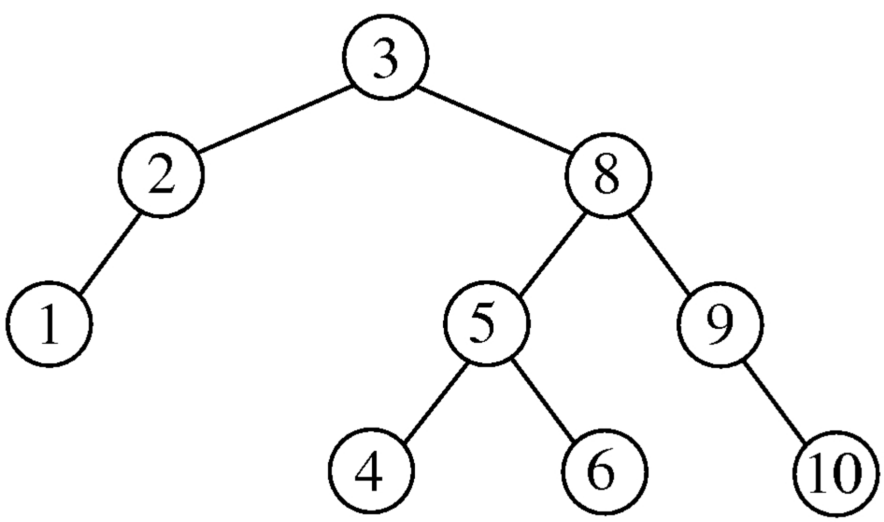
c. Fill in the blanks (i)-(iv) in the following routine for preorder traversal of a binary tree using a stack. Choose one of the following to fill in each blank:
`pop(S), pop(T), pop(T -> Left), pop(T -> Right),`
`push(T -> Left, S), push(T -> Right, S), push(T,S)`

Solution:
```
void Stack_Preorder (Tree T, Stack S) {
if (T == NULL) return; else push(T,S);
while (!isempty(S)) {
   T = pop(S);
   print_element(T -> Element);
   if  (T -> Right !=  NULL) push(T -> Right, S);
   if  (T -> Left !=  NULL) push(T -> Left, S);
   }
}
```

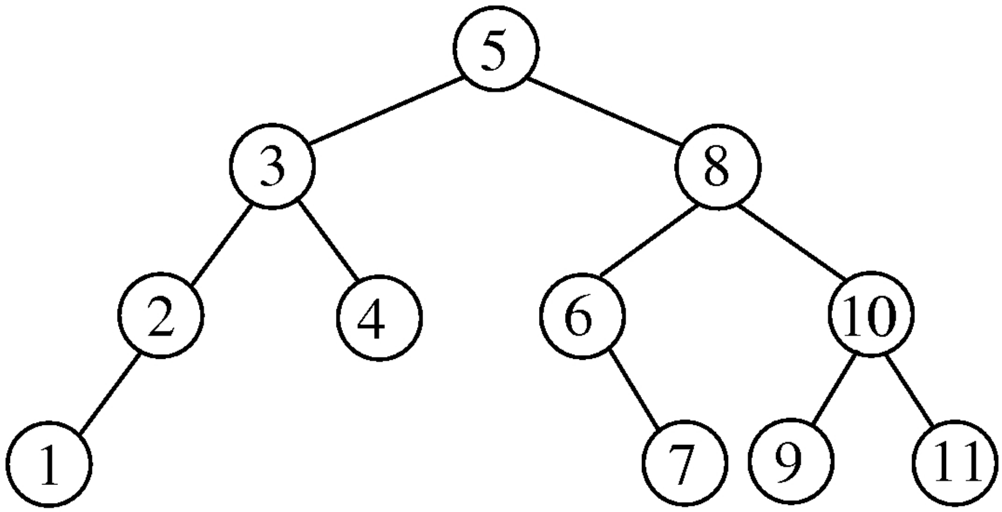
3. **Binary Search Trees**

a. What is the worst case running time for the Find operation on a tree of N nodes when you use: (i) an unbalanced binary search tree, (ii) an AVL tree, and (iii) a splay tree? Select one of the following for each: O(1), O(log N), O($\sqrt{N}$), O(N), O(N log N) (choose the best possible upper bound).
Solution: (i) O(N), (ii) O(log N), and (iii) O(N)

b. Draw the tree that results from inserting 11 followed by 7 into the following AVL tree:



Solution:



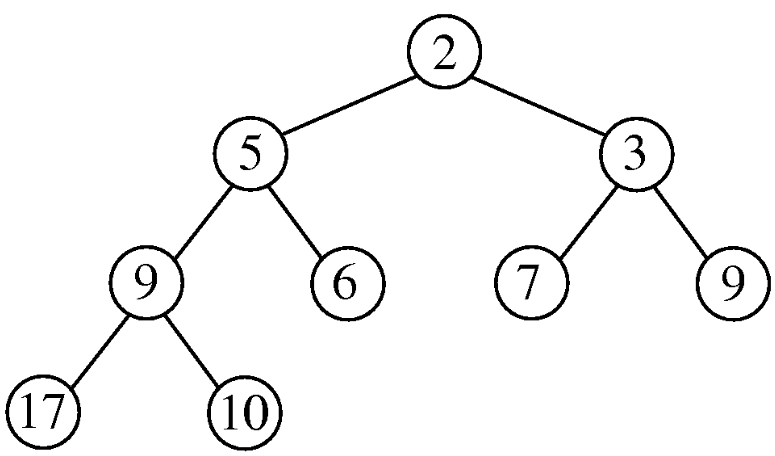
4. **Binary Heaps and Binomial Queues**

a. What are the two properties that make a binary tree a binary heap?

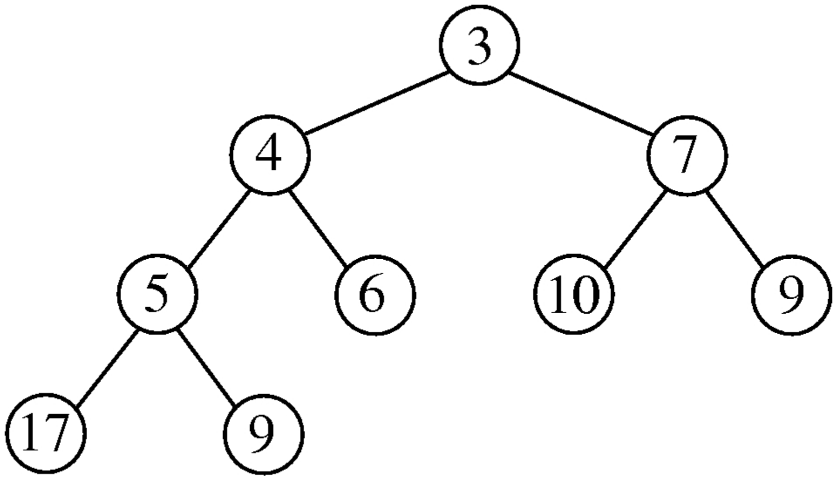
A binary heap is a binary tree that is:

1. Complete: all levels filled except possibly the bottom level, which is filled from left to right
2. Satisfies the heap order property: every node is smaller than (or equal to) its children

b. Draw the binary heap that results from deleting the minimum and then inserting 4 into the following binary heap:

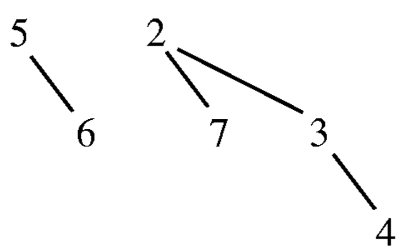


Solution:

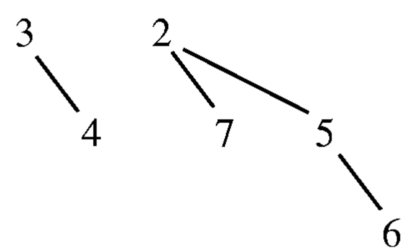


c. Draw the binomial queue that results from <u>inserting</u> the integers 1, 2, 3, 4, 5, 6, 7 (in that order) into an empty binomial queue and <u>then deleting the minimum</u>.

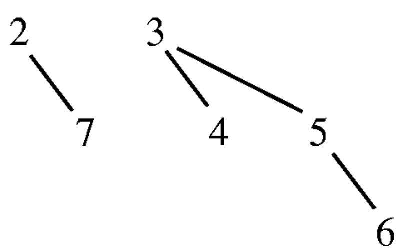
Solution: There are three possible solutions:



or

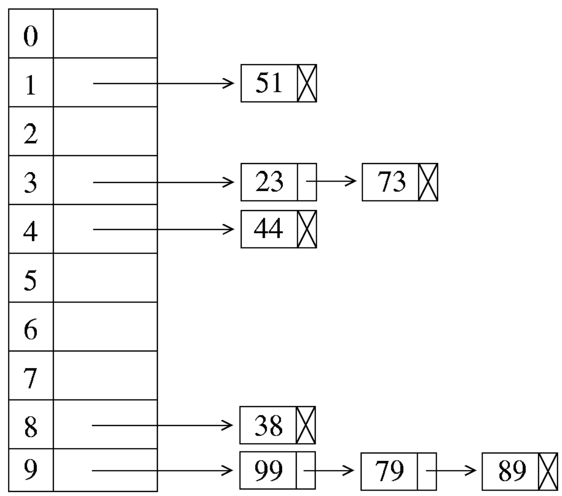


or



5. **Hashing**
Consider the hash function Hash(X) = X mod 10 and the ordered input sequence of keys 51, 23, 73, 99, 44, 79, 89, 38. Draw the result of <u>inserting</u> these keys in that order into a hash table of size 10 (cells indexed by 0, 1, …, 9).

a. separate chaining: (Note: 1. You may also insert new elements at the beginning of the list rather than the end; 2. You may also store the first element in the array and use a linked list for the second, third, … elements)



b. open addressing with linear probing, where F(i) = i;

0	79
1	51
2	89
3	23
4	73
5	44
6	
7	
8	38
9	99

c. open addressing with quadratic probing, where $F(i) = i^2$.

0	79
1	51
2	38
3	23
4	73
5	44
6	
7	
8	89
9	99