Birzeit University

Computer Science Dept.

Data Structures (Comp242/232/2321)

Midterm Exam

Student Name: Student No:

Instructors: Iyad Jaber, Mamoun Nawahdah, Radi Jarrar

Question #1 (30%):

A) Given a sorted array of distinct integers A[I,. . . ,n] the following algorithm find_i finds out whether there is an index I for which A[i] = I when calling on find_i(A.1.n).

Algorithm find_i (A, low, high)
Input: An array A of n distinct integers sorted in increasing order
output: An index I for which A[i] = i, null if such index cannot be found

```
If  low > high then
        return null //NOT FOUND
Else
    mid ←floor(high + Low)/2)
   if A[mid] = mid then
        return mid
else if A[mid] > mid then
        return find_i(A, low, mid-1)
 else
    return find_i(A, mid+1, high)
```

Write down the recurrence equation which describes the running time of algorithm find_i (A, low, high) as a function of n. Find the(Big Oh) complexity of Algorithm find_i(A, low, high).

Solution:

```
T(n) =    d              , n= 1   |
          T(n/2) + C     , n>1    |
                                  |
T(n) = T(n/2) + C                 |
T(n/2) = T(n/2²) + C              |
T(n) = T(n/2²) + 2C               |
T(n/4) = T(n/2³) + C              |
   --→  T(n) = T(n/2^k)  + kC     |
```

```
Let n/2^k = 1 -----→  n = 2^k --→ k = log n
T(n) = T(1) + C log n
     =  d + C log n

--→ T(n) = O (log n)
```

B) suppose we have an array based list A[0 ... n-1] and we want to delete all duplicates. Last position is initially n-1 , but gets smaller as elements are deleted. Consider the pseudo code program fragment in figure bellow. The procedure DELETE deletes the element in position j and collapses the list.

1) Rewrite this procedure using a linked list operations.
2) Using a standard array implementation**,** What is the running time of this procedure?
3) What is the running time using a linked list implementation?

```
for(int i = 0 ; i < last_Possition ; i++)
{
      int j = i + 1;
     while( j < last_Possition)
     {
        if( A[i] == A[j])
             Remove (j);
       else
             j++;
      }
}
```

Solutions :

1)

```
void function (List L){
            Node p = L ;
            Node t ;
          while (p != null) {
               t = p.next;
               k = p;
              while (t != null) {
                  if (p.element == t.element){
                     k.next = t.next;
                     t = k.next;
                   }
                  else{
                        k = t ;
                        t = t.next;
                        }
                   }
               p = p.next;
             }
      }
```

2) running time using array : $T(n) = O(n^2)$

3) running time using kinked list : $T(n) = O(n^2)$

Question #2 (20%):

Is it possible to implement the Stack ADT using one or more implementations of the Queue ADT ?
In other words, is it possible to implement the Stack ADT by using only one or more queues to hold the contents of the Stack? you may use only the Queue ADT methods of the queues to implement the methods specified in the Stack ADT. you may assume that the Queue ADT implementation exists that is complete and correct.

Write your answer below. if yes, describe in reasonable detail how such a Stack ADT implementation would be implemented, including at least a short explanation for each method in the ADT. if no, describe in reasonable detail why one or more Queue ADT implementations are not sufficient to implement the Stack ADT.

Solution :

YES, because the Stack implementation needs three methods (push, pop, top) which can be implemented using both methods of the Queue implementation (enque, deque) using two Queues, as following :

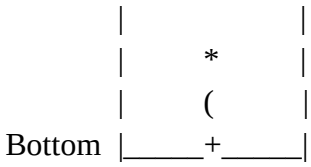
* push method can be implemented as the enque method for the first Queue (Q1)

* pop method can be implemented by the deque method for the first Queue (Q1) and returning the element

*top method can be implemented by the deque method for the first Queue (Q1) and enque method for the second Queue (Q2) and returning the top element of the second Queue (Q2).

Question #3 (25%):

A) consider the usual algorithm to convert an infix expression to postfix expression.
suppose that you have read 10 input characters during a conversion and that the stack now contains these symbols:



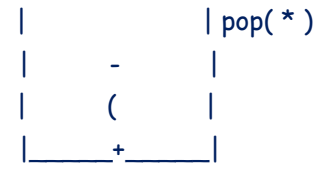
Now suppose that you read and process the 11th symbol of the input. Draw the stack for the case where 11th symbol is :

A Number :

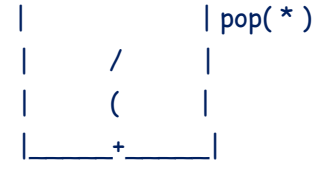
A Left parenthesis:

A right parenthesis:

A minus sign:



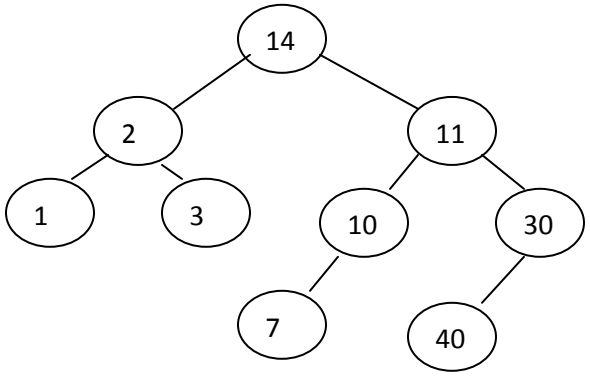
A division sign:



B) What is the value of the postfix expression ?

23, 4, /, 2, *, 12 , 3, -, +

C) consider this binary tree :



what is the order of nodes visited using a post-order traversal of the binary tree?

solution:

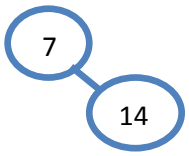
1, 3, 2, 7, 10, 40, 30, 11, 14

Question #4 (25%):

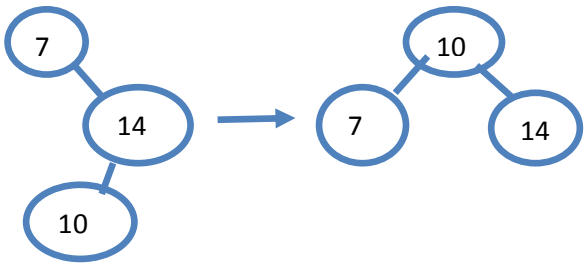
A) in the boxes below show the AVL trees result from the successive addition of the given values. Show the nodes, links and balance factors, Draw intermediate trees and clearly rotations, if any, and in what direction.

1. After adding 7 to an empty tree.

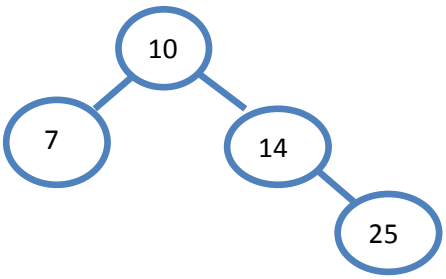
2. After adding 14 to the previous tree.



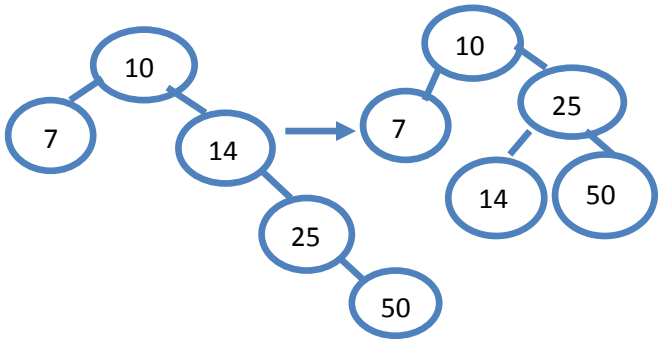
3. After adding 10 to the previous tree.



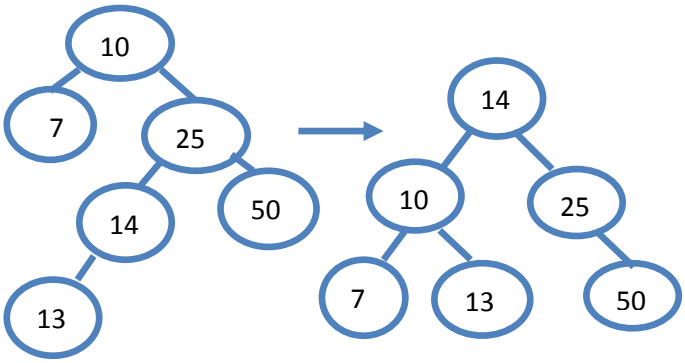
4. After adding 25 to the previous tree.



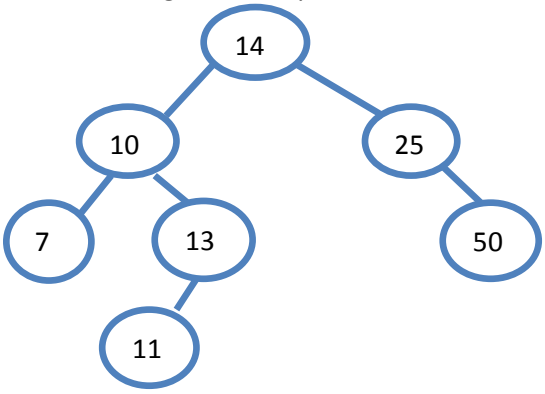
5. After adding 50 to the previous tree.



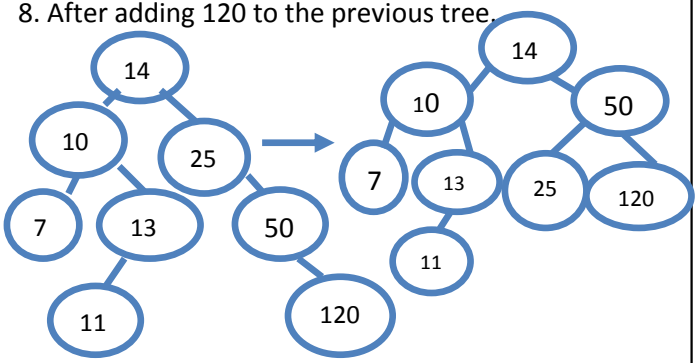
6. After adding 13 to the previous tree.



7. After adding 11 to the previous tree.



8. After adding 120 to the previous tree.



B) Write routine to list out the nodes of a binary tree in level-order. List the root, then nodes at depth 1, followed by nodes at depth 2, and so on, You must do this in linear time. Prove your time bound.

solution:

```
Queue Q ;
enque( Root ) ;
while ( ! Q.isEmpty( ) ){
     Node t = Q.deque();
     if ( t.left ! = null )
               Q.enque( t.left) ;
    if (t.right ! = null )
               Q.enque( t.right) ;
 }
```