Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

L'Université canadienne
Canada's university

CSI2110
Data Structures and Algorithms

Midterm Examination

Length of Examination: 2 hours — Oct 30, 2016, 3:00-5:00PM
Professors: M. Hoda, L. Moura, Y. Sami —

Last name: _____

First name: _____

Student number: _____

Signature: _____

Page	Marks of each page
PAGE 2	out of 5
PAGE 3	out of 5
PAGE 4	out of 3
PAGE 5	out of 3
PAGE 6	out of 5
PAGE 7	out of 1
PAGE 8	out of 3
PAGE 9	out of 2
PAGE 10	out of 3
TOTAL	out of 30

Closed Book.

Please answer in the space provided (in this questionnaire).

No calculators or electronic devices are allowed.

At the end of the exam, when time is up:

- Stop working and turn your exam upside down.
- Remain silent.
- Do not move or speak until *all* exams have been picked up, and a TA or a Professor gives the go-ahead.

NOTE: Where a big-Oh characterization is asked, give the best possible one.

Question 1 [2 points] What is the worst case running time of the following algorithms (in big-Oh notation) as a function of the size n of the array `a`?

```
public static int myCode1(int[] a) {   // node that a.length is always larger than 200
     int n=a.length;
     int total=0;
     for (int i=0; i<n; i++) {
            for (int j=0; j<200; j++) {
                  for (int k=0; k<j; k++) {
                        total += a[k]+i*j;
                  }
            }
     }
     return total;
}
```

a) $O(\log n)$ b) $O(n)$ c) $O(n^2)$ d) $O(n^3)$

```
public static int myCode2(int[] a) {
     int n=a.length;
     int total=0;
     for (int i=0; i<n; i++) {
            for (int j=0; j<200; j++) {
                  for (int k=0; k<i; k++) {
                        total += a[k]+i*j;
                  }
            }
     }
     return total;
}
```

a) $O(\log n)$ b) $O(n)$ c) $O(n^2)$ d) $O(n^3)$

Question 2 [3 points] Fill the blanks below:

- $900 + 100n^2 + 3n^5$ is $O($ $)$
- $\sum_{i=0}^{n} \frac{1}{2^i}$ is $O($ $)$
- $n + \log n^n$ is $O($ $)$

Question 3 [2 points]

(A) Consider elements with the following keys 50, 10, 2, 90, 3 that are inserted in this order into a **priority queue** where **maximum** keys have higher priorities. After that the elements are, one by one, removed from the priority queue with the appropriate operation and inserted into a **stack**. What will be the contents of the stack? (please list from bottom to top).

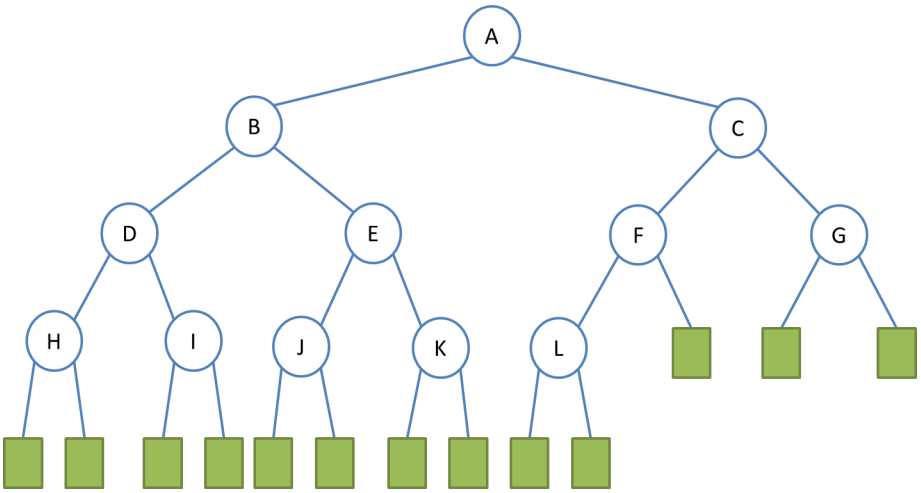
Final answer: (bottom)_____ (top)

(B) Considering your answer in part (A) the key returned by the first pop operation is:

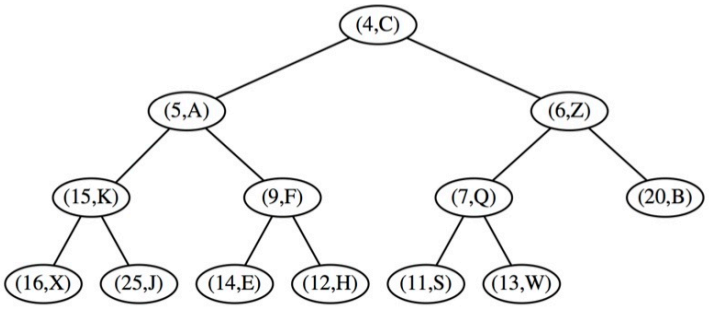
a) 50 b) 10 c) 2 d) 90 e) 3

Question 4 [3 points] List the nodes (by specifying the contained character) in the order they are visited for the binary tree shown below during:

- A Pre-order traversal:_____
- A Post-order traversal:_____
- An In-order traversal:_____



Question 5 [3 points]
Consider the following **min-heap** where the key is the integer displayed.



a) Show the heap after the operation `removeMin`

b) Starting from the original heap given in the question (do not use your heap obtained in part a), show the heap after inserting (1,L).

Question 6 [3 points] Below you are given an array with 6 keys which needs to be transformed into a **max-heap** via the **in-place** bottom-up heap construction algorithm seen in class.
Show the steps of this method, by showing at each step:

- the position/node we need to apply the next `downHeap` operation.
- the array after each `downHeap` operation.

3	9	5	1	12	21

Question 7 [5 points]

a) What is the number of nodes in a in a perfect binary tree of height h?
Answer:

Briefly justify with words or drawing.

b) What is the **minimum** number of **leaves** in a **full binary tree** with height 3?
Answer:

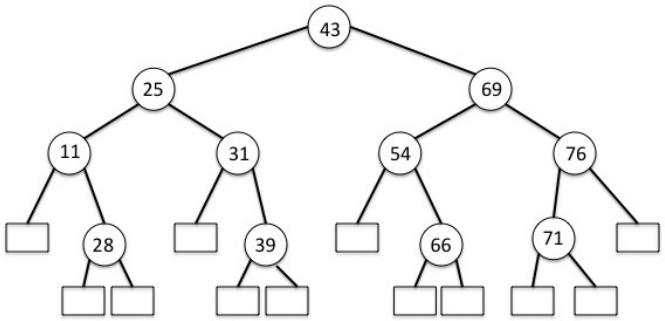
Briefly justify with words or drawing.

c) What is the big-Oh complexity of a delete operation in a binary search tree with N nodes that is perfect?

d) What is the big-Oh complexity of a delete operation in a binary search tree with N nodes such that each internal node has exactly one child?
(Note: if using dummy leaves the statement above about having one child is referring to child nodes that contain keys).

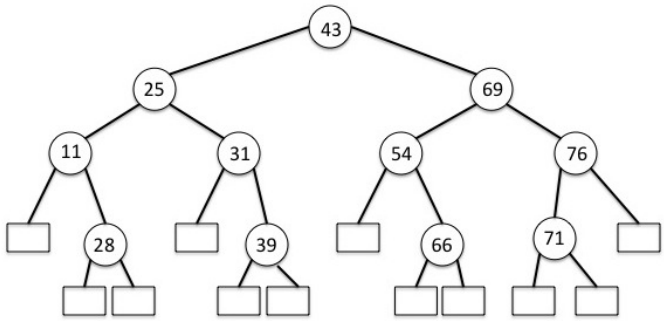
e) Consider a **min-heap** containing 7 elements and suppose we apply the **removeMin** operation.
What is the maximum number of swaps to perform this operation? _____
Give an example that achieves this maximum number of swaps:

Question 8 [1 points] For this question consider the following **binary search tree** and show the tree after deleting node 25.



Question 9 [3 points]

In this question you will insert key 74 in the following **AVL tree** which uses dummy leaves.



a) Show the tree after the first part of the insertion, in which the insertion is done as in a regular binary search tree.

b) In the tree you drew in part a), show nodes x, y, z and subtrees T_1, T_2, T_3 and T_4 to be used in the rebalancing step of the AVL tree insertion.

c) Show the AVL tree after the rebalancing step.

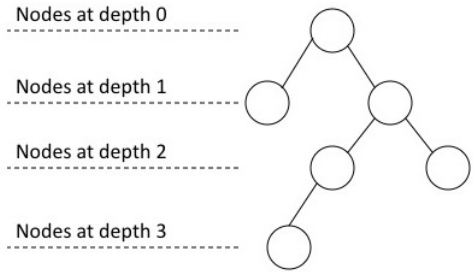
Question 10 [2 points] Give a $\Theta(n)$-time non recursive algorithm that reverses a singly linked list of n elements. The algorithm should use no more than constant storage beyond that needed for the list itself.
Note: The beginning of the linked list is pointed by variable `head` and each node in the list has fields: `element` (storing information) and `next` pointing to the next element in the list.

Example:
If before the algorithm the elements are stored in the list in the following order:
A, X, F, D, E
then after the algorithm is executed the elements will be stored in the following order:
E, D, F, X, A

Question 11 [3 points] Consider a **binary tree** Abstract Data Type that provides the following methods:

method	Description
root()	returns the position of the root of the tree or null if the tree is empty
parent(p)	returns the position of the node that is a parent of node p or null if p is the root
leftChild(p)	returns the position of the left child of p or null if p has no left child.
rightChild(p)	returns the position of the right child of p or null if p has no right child.
isInternal(p)	returns true if and only if p is an internal node.
isExternal(p)	returns true if and only if p is an external node.
size()	returns the number of nodes stored in the tree.

The **path length** of a binary tree T is the sum of the depths of all positions in T.
For example, the path length of the tree below is 9=0+1+1+2+2+3



Write an $O(n)$ algorithm that computes the path length of a binary tree T with n nodes.
You may give the required algorithms in pseudocode or Java program excerpt.
You may use the methods listed above and design any auxiliary method you wish.

(blank page to continue answers)