Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

CSI2110
Data Structures and Algorithms

Midterm Examination

Length of Examination: 2 hours — Oct 19, 2014, 15:00
Professor: L. Moura — Page 1 of 11

**You'll get marks for your best 50 points out of the 60 points available.
Feel free to answer all 60 points.**

Last name: ________________

First name: ________________

Student number: ________________

Signature: ________________

Page	Marks of each page
PAGE 2	out of 5
PAGE 3	out of 7
PAGE 4	out of 12
PAGE 5	out of 5
PAGE 6	out of 11
PAGE 7	out of 6
PAGE 8	out of 2
PAGE 9	out of 4
PAGE 10	out of 8
TOTAL	out of 60

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: In all questions where a big-Oh characterization is asked, please give the best possible one.

Question 1 [2 points]What is the worst-case running time of the following algorithms (in big-Oh notation) for input being a bi-dimensional $n \times n$ array A, for an arbitrary integer $n \geq 200$?

Algorithm ALG1(A, n) {
$tot \leftarrow 1$;
for $i \leftarrow 0$ **to** $n-1$ **do**
    for $j \leftarrow 0$ **to** 100 **do**
        for $k \leftarrow 0$ **to** $j * j$ **do**
            $tot \leftarrow tot + A[i][j] * k$;
return tot;
}

a) $O(1)$ b) $O(n)$ c) $O(n^2)$ d) $O(n^3)$ e) $O(n^4)$

Algorithm ALG2(A, n){
$tot \leftarrow 1$;
for $i \leftarrow 0$ **to** $n-1$ **do**
    for $j \leftarrow 0$ **to** 100 **do**
        for $k \leftarrow 0$ **to** $i * i$ **do**
            $tot \leftarrow tot + A[i][j] * k$;
return tot;
}

a) $O(1)$ b) $O(n)$ c) $O(n^2)$ d) $O(n^3)$ e) $O(n^4)$

Question 2 [3 points] Fill the blanks below:

- $7\sqrt{n} + 10\log_2 n^2$ is $O($ $)$
- $(\sum_{i=0}^{100}(i \cdot n^3))$ is $O($ $)$
- $n^4 + 10n^2$ is $O($ $)$

Question 3 [4 points]
Consider the following part of a recursive program in Java describing the following method of a general tree (not necessarily binary) that stores integer numbers:

```
public int Calc(Position<int> p) {
int a=p.getElement();
for(Position<int> c: children(p))
   a = a + Calc(c);
return a;
}
```

A. What is the result of `Calc(v)`, where v is the root of the tree?

B. What is the worst-case running time in big-Oh notation of `Calc(v)`, where `v` is the root of a tree with n elements?

Question 4 [3 points] Prove that $f(n) = n^2 + \log_2 n^4$ is $O(n^2)$ by using the definition of big-Oh.

Question 5 [3 points] Complete the following sentences with the tightest big-Oh estimate for the worst-case running time of the corresponding algorithms. The number of elements in each data structure is an arbitrary number n.

1. Method `insert` of a heap: $O(\qquad)$
2. Method `push` of a Stack implemented with an array: $O(\qquad)$
3. Method `push` of a Stack implemented with an extendible array: $O(\qquad)$
4. Method `enqueue` of a Queue implemented with a doubly linked list: $O(\qquad)$
5. Method `removelast` of a Deque (double-ended queue) implemented with a doubly linked list: $O(\qquad)$
6. Algorithm `heapSort` on an array of n elements: $O(\qquad)$

Question 6 [5 points] Short answers:

- The array that represents a **min-heap** is always sorted. [TRUE/FALSE]
- The height of a **full** binary tree of n nodes is $O(\log n)$.[TRUE/FALSE]
- The height of a **complete** binary tree of n nodes is $O(\log n)$.[TRUE/FALSE]
- The **maximum** number of nodes in a binary tree of height h is ______
- The **minimum** number of nodes in a binary tree of height h is ______

Question 7 [2 points] Let h_T be the height of a **full binary tree** T with 9 nodes. Let i and e be its number of internal and external nodes, respectively.

- The number of internal nodes of T is $i =$ ______
- The number of external nodes of T is $e =$ ______
- The minimum possible value for h_T is ______
- The maximum possible value for h_T is ______

Question 8 [2 points] Which of the following data structures can **always** be searched for its minimum element in $O(1)$ time (multiple answers are possible)?

a) unsorted array
b) sorted array
c) sorted doubly-linked list
d) min-heap

Question 9 [2 points] Consider a Queue implemented using a circular array, with the contents given below:

```
Front = 0
Rear = 4
```

index i	0	1	2	3	4
value $Q[i]$	Z	T	X	Y	

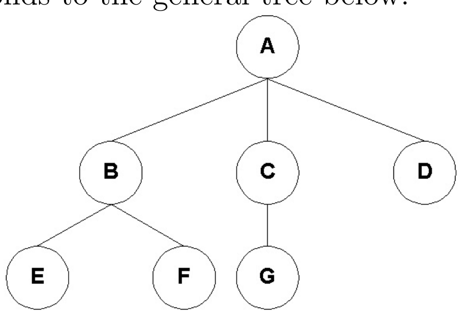
Show the data structure after the following operations are executed in the given order:

Dequeue(); Enqueue(B); Dequeue(); Enqueue(A);

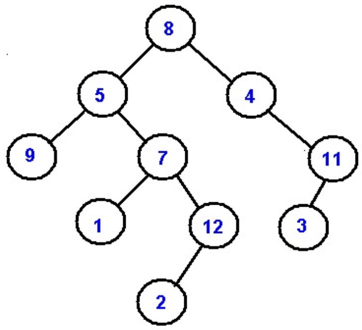
```
Front = ________
Rear = ________
```

index i	0	1	2	3	4
value $Q[i]$					

Question 10 [3 points] Using the algorithm seen in class, construct the binary tree that corresponds to the general tree below:



Question 11 [3 points] Consider the following binary tree.



List the numbers stored in the nodes in the following order traversals:

- Pre-order:
- Post-order:
- In-order:

Question 12 [4 points] Consider the following complete binary tree using an array representation. This array needs to be transformed into a **max-heap** via the ***bottom-up*** **Heap construction** algorithm seen in class.

index i	1	2	3	4	5	6	7	8	9
value $A[i]$	9	7	12	10	11	30	8	14	20

Show the array after *bottom-up* Heap construction algorithm has been applied. (Use back of pages as draft to simulate this algorithm.)

index i	1	2	3	4	5	6	7	8	9
value $A[i]$									

Question 13 [4 points] Consider the **in-place Heapsort algorithm**. After the first phase of this algorithm (build heap), the array is as follows:

index i	1	2	3	4	5	6
value $A[i]$	12	11	9	8	5	7

Show the state of the array A after the first 2 `removeMax()` operations in the execution of Heapsort.

index i	1	2	3	4	5	6
$A[i]$ after first removeMax()						
$A[i]$ after second removeMax()						

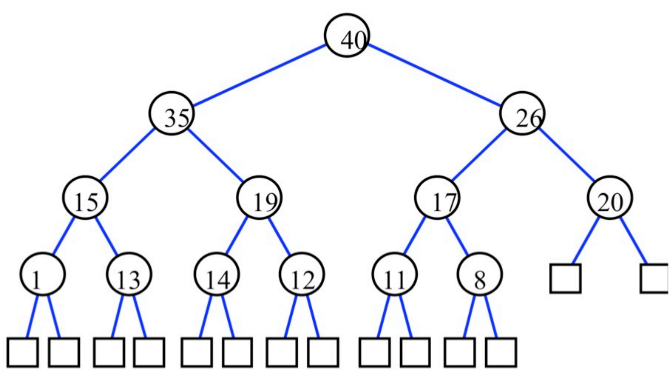
Question 14 [4 points] We want to sort in increasing order the following sequence S, implemented using an array A, using **in-place Insertion Sort** .

index i	1	2	3	4	5	6	7	8	9	10
value $A[i]$	9	7	3	8	4	12	5	15	2	11

The array will be composed of a sorted part (growing after each step) followed by an unsorted part (shrinking after each step). **Provide the contents** of the array A after the $3rd$ and $4th$ steps of the sort (i.e., after inserting the $3rd$ and the $4th$ element), and **circle the sorted part** of the array. Of course, you need to first simulate the $1st$ and $2nd$ steps using draft space, but they do not need to be shown here.

index i	1	2	3	4	5	6	7	8	9	10
$A[i]$ after step 3										
$A[i]$ after step 4										

Question 15 [2 points] Consider the following max-heap:



Show the heap after `insert(37)`:

Question 16 [2 points] Consider the abstract data type `Deque` (double-ended queue) with the following methods:

- `void insertFirst(E e)`: insert `e` at the beginning of the Deque.
- `void insertLast(E e)`: insert `e` at the end of the Deque.
- `E removeFirst()` removes and returns the first element of the Deque; an error occurs if the Deque is empty.
- `E removeLast()` removes and returns the last element of the Deque; an error occurs if the Deque is empty.
- `E first()`: returns the first element of the Deque; an error occurs if the Deque is empty.
- `E last()`: returns the last element of the Deque; an error occurs if the Deque is empty.
- `int size()`: returns the number of elements stored in the Deque.
- `boolean isEmpty()`: returns true if and only if the Deque is empty.

Adapt the `Deque` as a `Stack` abstract data type and fill in the table below:

Stack method	Deque Method
`void push(E e)`	
`E pop()`	
`boolean isEmpty()`	
`E top()`	

Question 17 [4 points]

Consider the **array representation** of a **complete** binary tree where the nodes of the tree are represented by positions $1, 2, \cdots, n$ of an array A which stores its elements. Write a method "`void rightmostDescendant(int i)`" that swaps the element stored at position `i` of the array A with the element of its right-most descendant in the tree.

You may give your algorithm in pseudocode or Java-like program.

Question 18 The following two parts consider a **binary tree** Abstract Data Type that provides the following methods:

method	Description
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.

For parts A and B, you may give the required algorithms in pseudocode or Java program excerpt. You may use the methods listed above.

A. [4 points] Design an algorithm for the following operation of a binary tree T: `inorderNext(q)`: return the position visited after `q` in an in-order traversal of T (or `null` if `q` is the last node to be visited).

B. [4 points] Write an $O(n)$ algorithm that returns the number of external nodes in a binary tree T with n nodes. Your algorithm will be invoked by calling `numExternal(r)`, where `r` is the root node of the tree.
Hint: You may implement a recursive algorithm.

(blank page to continue answers)