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 Fall 2017

Midterm Exam

October 29, 15h00 - 120 minutes

30 points (30% of the final mark)

Closed book

LAST NAME: SOLUTION

First Name: _____ Student Number: _____

Signature _____

Instructions :

Closed book exam. No calculators allowed. There are 11 pages in this exam.

In all the questions, when asked for a big-Oh, please provide the best possible value.

All the logarithms are given in base 2.

QUESTION	MARKS OBTAINED
Questions 1-9 (multiple choice)	/9
Question 10	/3
Question 11	/3
Question 12	/2
Question 13	/5
Question 14	/5
Question 15	/3
TOTAL	/30

The following text refers to questions 1-3 :

The method `process(data,k)` processes the first `k` elements of an array containing `N` elements (`k<N`) and this method has a complexity of O(log k).

What will be the complexity of the following algorithms/pieces of code?

Question 1 [1 point]

```
for (int i=1; i<=N; i++) {
     process(data, i);
   }
```

LOOP RUN N TIMES

$\Theta(\log i) \leq \Theta(\log N)$

A) O(1) B) O(log N) C) O(N) D) O($(\log N)^2$) E) O(N log N)

Question 2 [1 point]

```
for (int i=1; i<=N; i++) {
   process(data, N);
 }
```

loop RUN N TIMES

$\Theta(\log N)$

A) O(1) B) O(log N) C) O(N) D) O($(\log N)^2$) E) O(N log N)

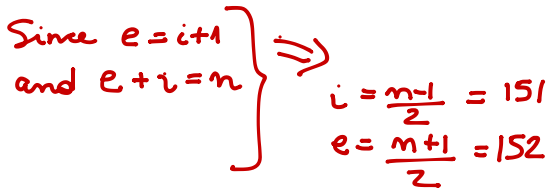
Question 3 [1 point]

```
process(data, N/4);
process(data, N/2);
```

$\Theta(\log N/4)$
$\Theta(\log N/2)$
$\Theta(\log N)$

A) O(1) B) O(log N) C) O(N) D) O($(\log N)^2$) E) O(N log N)

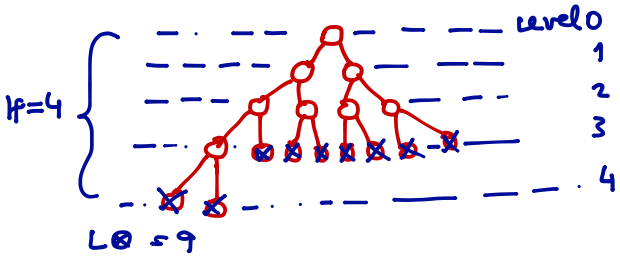
Question 4 [1 point] Recall that a full binary tree is a rooted tree in which every internal node has exactly two children. What is the number of internal nodes and external nodes in a full binary tree with 303 nodes?



A) 101 internal nodes and 202 external nodes
B) 152 internal nodes and 151 external nodes
C) 202 internal nodes and 101 external nodes
D) 151 internal nodes and 152 external nodes
E) the number of internal and external nodes cannot be determined from the number of nodes only.

Question 5 [1 point]
The height H and the number of leaves L of a complete binary tree with 17 nodes are:

A) H=4 L=2
B) H=4 L=9
C) H=4 L=16
D) H=5 L=2
E) None of the above.

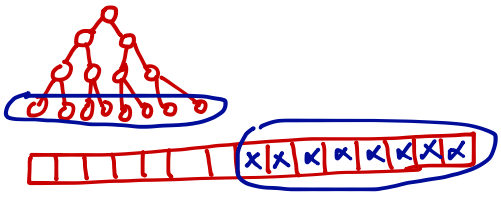


Question 6 [1 point]
Consider a **priority queue** with `n` elements, implemented using an **unsorted array**.
Which of the following answers give the worst-case running time of the operations
`insert` (insertion of an arbitrary element into the priority queue)
`removeMin` (deletion of the smallest element in the priority queue)
`min` (return the smallest element without modifying the priority queue), respectively:

A) Θ (1), Θ (1), Θ (1)
B) Θ (1), Θ (n), Θ (1)
C) Θ (1), Θ (n), Θ (n)
D) Θ (n), Θ (log n), Θ (log n)
E) None of the above.

Question 7 [1 point]
A **perfect binary tree** made of 15 nodes is implemented with an **array**. If we wish to find out if value 17 is stored in a leaf of this tree, how many nodes have to be visited?

A) 15
B) 7
C) 8
D) 2
E) None of the above.



Question 8 [1 point]

In which of the following data structures containing `n` elements, the worst case complexity of searching for an element with a given key k is `O(log n)`?

A) a heap
B) a binary search tree
C) a sorted array
D) a queue
E) more than answer above.

Question 9 [1 point] Consider the following Java code of a well known sorting method

```
public static void mySort( int [ ] array ) {
    public static void insertionSort(int array[]) {
        int n = array.length;
        for (int j = 1; j < n; j++) {
            int key = array[j];
            int i = j-1;
            while ( (i >= 0) && ( array [i] > key ) ) {
                array [i+1] = array [i];
                i--;
            }
            array[i+1] = key;
        }
    }
```

Give the exact asymptotic complexity for the **best case** running time and **worst case** running time of `mySort` as a function of the number of elements `n` of the input array:

A) best case: $\Theta(n^2)$, worst case: $\Theta(n^2)$
B) best case: $\Theta(n)$, worst case: $\Theta(n \log n)$
C) best case: $\Theta(\log n)$, worst case: $\Theta(n \log n)$
D) best case: $\Theta(n)$, worst case: $\Theta(n^2)$
E) none of the above

Question 10 [3 points] Give an asymptotic upper bound using **big-Oh** for the following functions. Use the tightest and simplest upper bound. Briefly justify your answers.

a) $f(n)= 2n^3+2^3$ $\leq 2n^3+8 \leq 2n^3+8n^3 \leq 10n^3 \quad \forall n \geq 1$

$O(n^3)$

b) $f(n)= \log(n)+\log(n/2)$ $\leq 2 \log n \quad \forall n \geq 1$

$O(\log n)$

c) $f(n)= \log(n^2)$ $\leq 2 \log n \quad \forall n \geq 1$

$O(\log n)$

d) $f(n)=n \sum_{i=1}^{n} 1/2^i$ $= n\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\ldots+\frac{1}{2^n}\right)$

$\leq n \cdot 1$ $O(n)$

e) $f(n)= (n+n^2)\log(n)$ $\leq (n^2+n^2)\log n = 2n^2 \log n$

$O(n^2 \log n)$

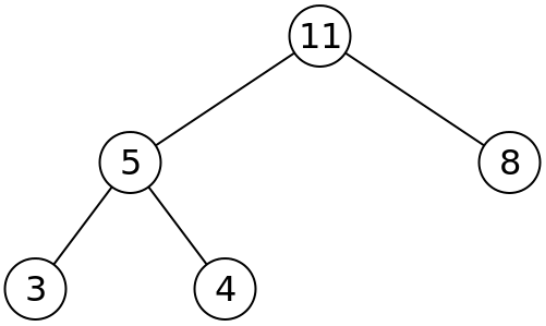
f) $f(n)=\sum_{i=1}^{n}(n-i) \leq \sum_{i=1}^{n} n = n^2$

OR

$= \sum_{j=0}^{n-1} j = \frac{(n-1)(n)}{2} = \frac{n^2-n}{2} \leq \frac{n^2}{2}$

$O(n^2)$

Question 11 [3 points] Consider the following max-heap:



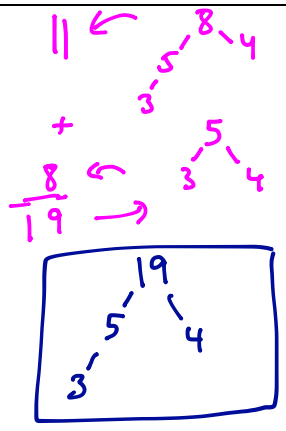
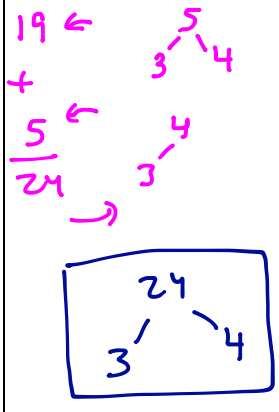
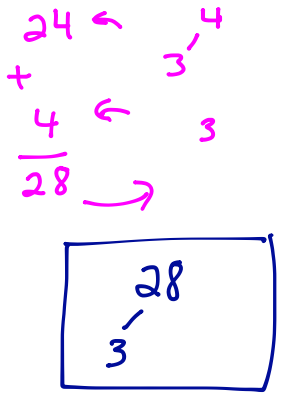
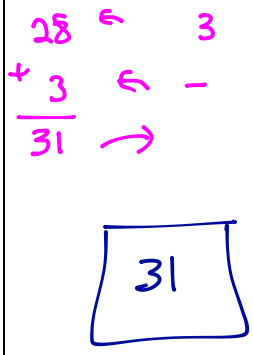
and the following operations :

`insert(key)` : insert a node with the value key in the heap

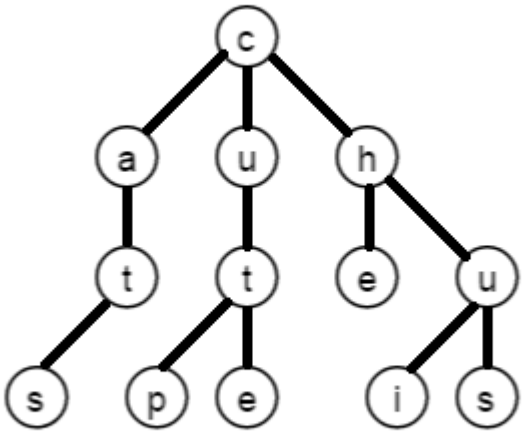
`removeMax()` : delete and return the element of maximum key in the heap

What will be the heap produced by the following algorithm? Draw the heap at the end of each loop iteration

```
for (int i=0 ; i<4; i++) {
   insert (removeMax() + removeMax());
   //**show the heap at this point of the code at each iteration
}
```

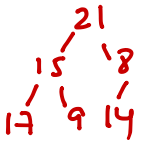
i=0	i=1	i=2	i=3
			

Question 12 [2 points] Print the pre-order and post-order traversal of the following tree:



pre-order : c a t s u t p e h e u i s

post-order: s t a p e t u e i s u h c



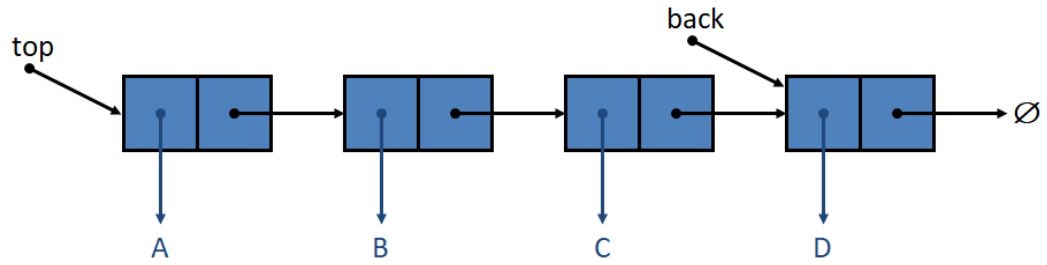
Question 13 [5 points] Given the following array:

	21	15	8	17	9	14

Sort this array in-place in increasing order using the HeapSort algorithm. You must show the state of this array after each downheap operations
(You may need more or less arrays as the ones provided below)

	21	15	14	17	9	8
	21	17	14	15	9	8
	21	17	14	15	9	8
	17	15	14	8	9	21
	15	9	14	8	17	21
	14	9	8	15	17	21
	9	8	14	15	17	21
	8	9	14	15	17	21

Question 14 [5 points]
A **stack** is implemented using a linked list.



The pseudo-code of this implementation is as follows (note the extra back variable pointing to the last element of the list):

```
Algorithm push(obj):
   n ← new Node
   n.item ← obj
   n.setNext(top)
   top ← n
   if size = 0
      back ← top
   size++
```

```
Algorithm pop():
  if isEmpty() then
        ERROR
  temp ← top.item
  top ← top.getNext()
  size--
  if size = 0
      back ← top
  return temp
```

The empty stack is initialized by doing: `back ← null; top ← null.`

Now, we want to implement a **queue** using the same linked list implementation.
Knowing that the `dequeue()` function has been implemented as follows:

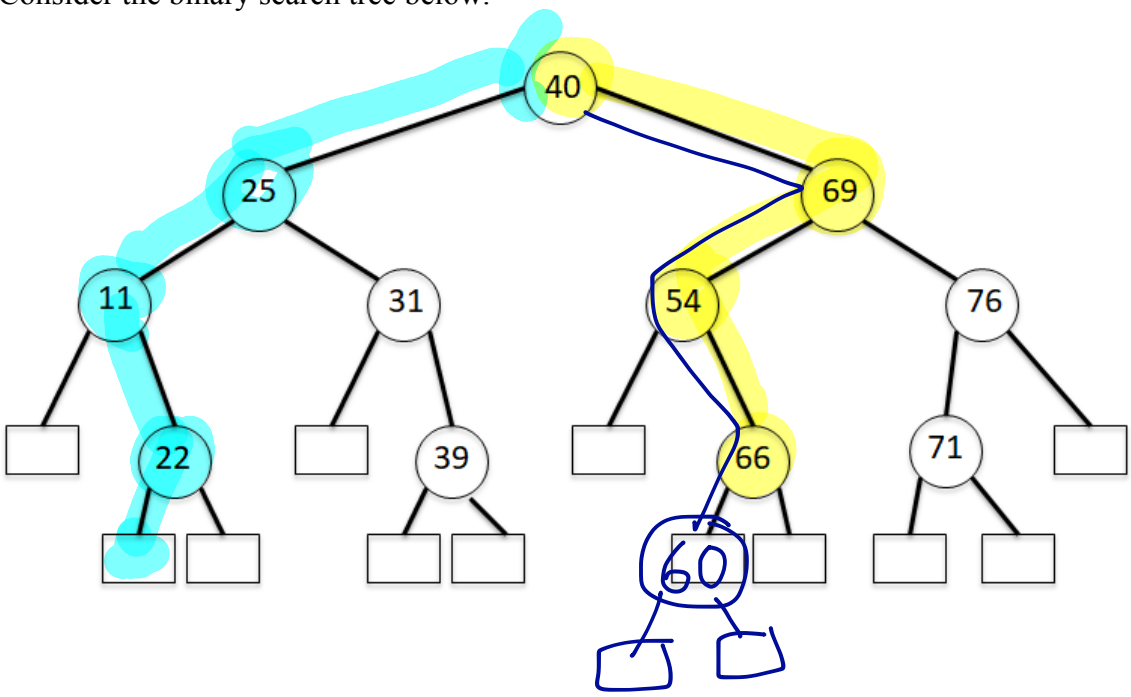
```
Algorithm dequeue():
  return pop()
```

Write the pseudo-code of the algorithm `enqueue`. <u>This method should run in time O(1)</u> :

```
Algorithm enqueue(obj):
   n ← new Node
   n.item ← obj
   if (size = 0) { back ← n
                   top ← n }
   else { back.setNext(n)
          back ← n
        }
```

Question 15 (3 points)

Consider the binary search tree below.



a) How many comparisons are done when searching for key 66 ?
List all the keys that are compared.

4 comparisons, 40, 69, 54, 66

b) How many comparisons are done when searching for key 15 ?
List all the keys that are compared.

5 comparisons 40, 25, 11, 22, NULL
(accept if say 4 comparisons)

c) Insert a new node with key 60 and show the resulting tree.
(you may to draw on top of the given tree as long as it is legible)

(page intentionally left blank)