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

CSI2510 Automne 2017

Examen de mi-session

Le 29 octobre à 15h00 120 minutes

30 points (30% de votre note finale)

Aucune documentation permise

Nom: _____

Prénom: _____ No étudiant: _____

Signature _____

Dans toutes les questions où la notation grand 'O' est demandée, donner la meilleur valeur possible.

Tous les logarithmes sont en base 2.

QUESTION	POINTS
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

La fonction `process(data, k)` permet le traitement des `k` premiers éléments du tableau `data` contenant `N` valeurs (`k<N`). Cette fonction a une complexité O(log k).

Quelle sera donc la complexité des algorithmes suivants ? (Questions 1 à 3)

Question 1 [1 point]

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

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);
 }
```

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);
```

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

Question 4 [1 point]

Un arbre binaire plein est un arbre binaire pour lequel tous les nœuds internes ont exactement 2 enfants. Donc un arbre plein fait de 303 nœuds aura :

A) 101 noeuds internes et 202 noeuds externes
B) 152 noeuds internes et 151 noeuds externes
C) 202 noeuds internes et 101 noeuds externes
D) 151 noeuds internes et 152 noeuds externes
E) Le nombre exact de noeuds internes et externes ne peut être determiné à partir du nombre total de noeuds.

Question 5 [1 point]

La hauteur H et le nombre de feuilles L d'un arbre binaire complet composé de 17 noeuds sont:

A) H=4 L=2
B) H=4 L=9
C) H=4 L=16
D) H=5 L=2
E) Aucune de ces réponses

Question 6 [1 point]

Soit une file à priorité contenant `n` éléments réalisée avec un tableau non-trié. Laquelle des réponses suivantes donne la complexité respective, au pire cas, des opérations `insert` (insertion d'un élément dans la file), `removeMin` (retrait du plus petit élément) et `min` (retour du plus petit élément sans modifier la file):

A) $\Theta(1)$, $\Theta(1)$, $\Theta(1)$
B) $\Theta(1)$, $\Theta(n)$, $\Theta(1)$
C) $\Theta(1)$, $\Theta(n)$, $\Theta(n)$
D) $\Theta(n)$, $\Theta(\log n)$, $\Theta(\log n)$
E) Aucune de ces réponses

Question 7 [1 point]

Un arbre parfait de 15 nœuds est réalisé avec un tableau. Si nous voulons savoir si la valeur 17 est contenu dans l'une de ses feuilles, combien d'éléments faut-il visiter ?

A) 15
B) 7
C) 8
D) 2
E) Aucune de ces réponses

Question 8 [1 point]

Dans laquelle de ces structures de données contenant `n` éléments, la complexité au pire cas pour obtenir le plus petit élément est égale à O(log n) :

A) un monceau (heap)
B) un arbre binaire de recherche
C) une liste triée
D) une file
E) Plusieurs de ces réponses

Question 9 [1 point]

Soit le code Java ci-dessous réalisant un algorithme de tri bien connu.

```
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;
        }
    }
```

Donner la complexité au <u>meilleur cas</u> et au <u>pire cas</u> de `MySort` en fonction de la taille `n` du tableau à trier:

A) meilleur cas: $\Theta(n^2)$, pire cas: $\Theta(n^2)$
B) meilleur cas: $\Theta(n)$, pire cas: $\Theta(n \log n)$
C) meilleur cas: $\Theta(\log n)$, pire cas: $\Theta(n \log n)$
D) meilleur cas: $\Theta(n)$, pire cas: $\Theta(n^2)$
E) Aucune de ces réponses

Question 10 [3 points] Quelle est la complexité Grand O (*big-Oh*) des fonctions ci-dessous? Justifier vos réponses par de courtes démonstrations :

a) f(n)= $2n^3+2^3$

b) f(n)= log(n)+log(n/2)

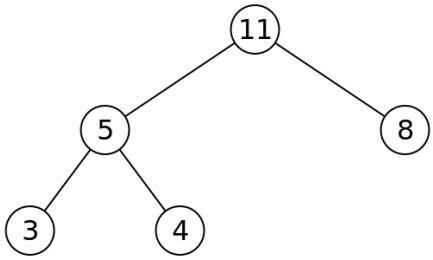
c) f(n)= $\log(n^2)$

d) f(n)=$n \sum_{i=1}^{n} {}^{1}/_{2^i}$

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

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

Question 11 [3 points] Soit l'arbre monceau suivant :



et les opérations suivantes effectuées sur cet arbre :

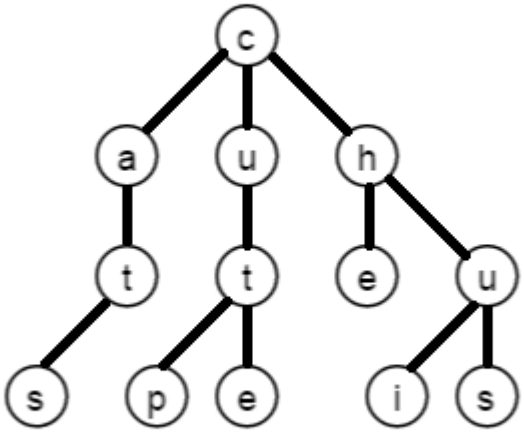
`insert(key)` : permet d'insérer un noeud ayant la valeur de la clé dans l'arbre
`removeMax` : retourne l'élément de clé maximum et l'enlève du monceau

Quel arbre produira l'algorithme suivant ? Dessiner l'arbre monceau à la fin de chaque itération

```
for (int i=0 ; i<4; i++) {
    insert (removeMax() + removeMax());
   // show the tree here
}
```

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

Question 12 [2 points] Déterminer la traversée pré-ordre et post-ordre de l'arbre suivant :



pré-ordre : _____

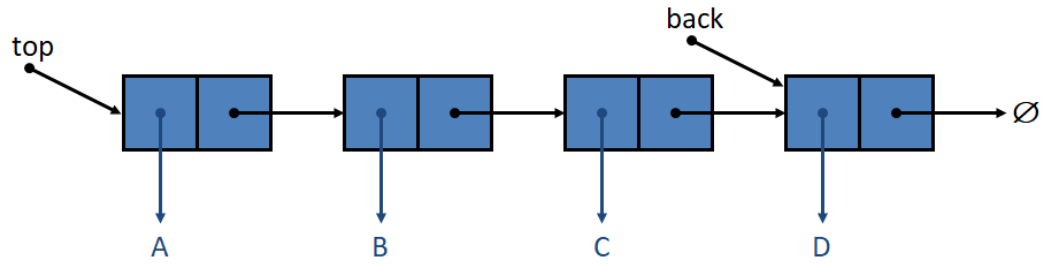
post-ordre: _____

Question 13 [5 points] Soit le tableau suivant :

	21	15	8	17	9	14

Trier ce tableau sur place en ordre croissant en utilisant l'algorithme du tri par monceau (*HeapSort*). Vous devez montrer l'état du tableau après chaque descente (downheap operations) de monceau. (vous pourriez avoir besoin de plus ou moins de tableau que ceux pré-dessinés ci-dessous)

Question 14 [5 points] Une pile est réalisée à l'aide d'une liste chainée.



Le pseudo-code pour cette pile est comme suit (noter la variable `back` pointant au dernier élément de la liste):

```
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
```

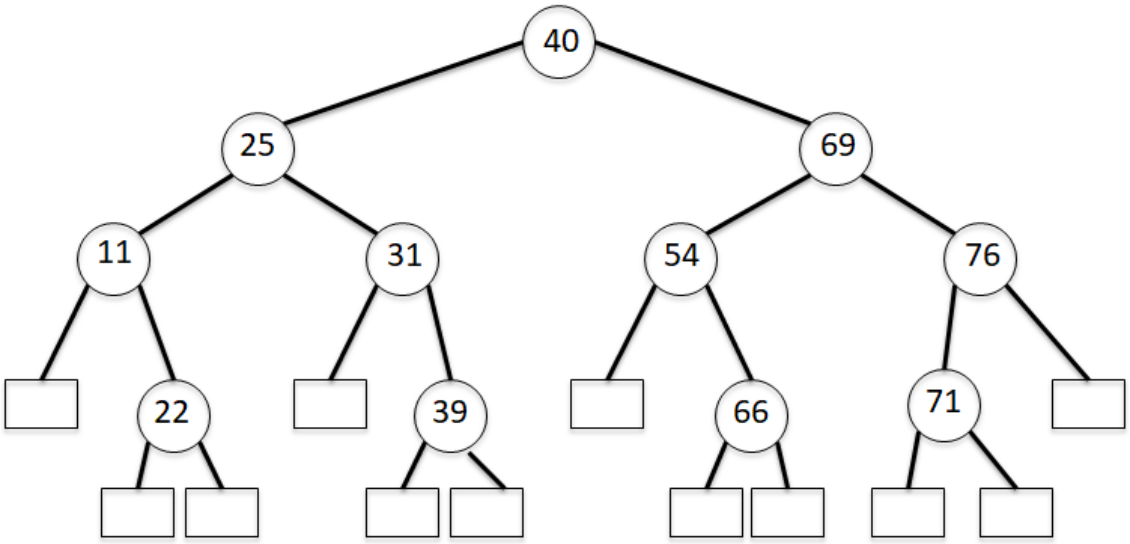
Initialement on a: `back ← null;  top ← null.`
Nous voulons maintenant réaliser une file avec la même liste chainée.
Sachant que la fonction `dequeue` qui enlève un objet à la file est réalisée comme suit :

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

Écrire le pseudo-code de la fonction `enqueue`. Cette fonction doit être de complexité O(1) :

Question 15 (3 points)

Soit l'arbre binaire de recherche ci-dessous.



a) Combien de comparaisons seront effectuées lorsque la clé 66 est recherchée? Lister les clés qui seront comparées.

b) Combien de comparaisons seront effectuées lorsque la clé 15 est recherchée? Lister les clés qui seront comparées.

c) Insérer un nouveau nœud dans cet arbre dont est 60. Vous pouvez dessiner votre réponse sur l'arbre montré.