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 2018

Examen Final

Prof. Lucia Moura et Prof. Robert Laganiere

10 Décembre, 14:00 – 3 heures

40 points (40%)

Nom de famille: _____

Prénom: _____

Numéro d'étudiant: _____

Signature _____

Instructions :
Aucune documentation permise. Aucune calculatrice.
Pour la complexité Grand O, toujours spécifier la meilleure valeur.
Tous les logarithmes sont en base 2.

QUESTION	MARKS OBTAINED
Q1-15 choix multiples	/15
Q16	/ 2
Q17	/ 2
Q18	/ 6
Q19	/ 4
Q20	/ 3
Q21	/ 2
Q22	/ 2
Q23	/ 1
Q24	/ 3
TOTAL	/40

Question 1 [1 point]
Soit la fonction suivante:

```
int mycode(int n) {
    int i, j, k = 0;
    for (i  = n/2; i <= n; i++)
        for (j = 2; j <= n; j = j * 2)
            k = k + 2;
    return k;
 }
```

La complexité Grand O de cette fonction est:

A)	$\Theta(n^2)$
B)	$\Theta(n^2 \log n)$
C)	$\Theta(n^3)$
D)	$\Theta(n^3 \log n)$
E)	Aucune de ces réponses

Question 2 [1 point]
Le petit programme ci-dessous manipule un graphe non-orienté G représenté à l'aide des listes d'adjacence.

```
int count=0;

// run through all vertices v of the graph
for each (v: G.vertices()) {
     // run through all edges incident to v
     for each (e: G.incidentEdges(v))
                  count++;
```

Soit `n` le nombre de sommets dans le graphes et `m` le nombre d'arêtes, alors la fonction décrivant le plus précisément la complexité en temps d'exécution de ce programme est:

A)	$O(n^3)$
B)	$O(n^2)$
C)	$O(n*m)$
D)	$O(n+m)$
E)	Aucune de ces réponses

Question 3 [1 point]
Lequel des énoncés suivants est FAUX?
A) $2^{\log n}$ is $O(2^n)$
B) n^2 is $O(2^n)$
C) $2^{\log n}$ is $O(n^2)$
D) n^2+2^n is $\Omega(n)$
E) Ils sont tous vrais.

Question 4 [1 point]
Une structure de données (monceau, arbre AVL, table de hachage) peut être utilisée pour réaliser un type abstrait de données (file à priorité, dictionnaire, file). Laquelle des affirmations suivantes est vraie.

A)
Le monceau (*Heap*) est souvent utilisé pour réaliser une file à priorité.
L'arbre AVL est souvent utilisé pour réaliser un dictionnaire.
Une table de hachage est souvent utilisée pour réaliser un dictionnaire (*Map*).

B)
Le monceau (*Heap*) est souvent utilisé pour réaliser une file à priorité.
L'arbre AVL est souvent utilisé pour réaliser une file à priorité.
Une table de hachage est souvent utilisée pour réaliser un dictionnaire (*Map*).

C)
Le monceau (*Heap*) est souvent utilisé pour réaliser un dictionnaire (*Map*).
L'arbre AVL est souvent utilisé pour réaliser un dictionnaire (*Map*).
Une table de hachage est souvent utilisée pour réaliser une file.

D)
Le monceau (*Heap*) est souvent utilisé pour réaliser une file.
L'arbre AVL est souvent utilisé pour réaliser une file à priorité.
Une table de hachage est souvent utilisée pour réaliser un dictionnaire (*Map*).

E) Aucune de ces réponses

Question 5 [1 point]
Compléter la phrase suivante:
Le pire cas, en temps d'exécution, de la recherche d'une clé dans un arbre AVL est (i) _____ alors que le pire cas, en temps d'exécution, de la recherche d'une clé dans un arbre de recherche régulier est (ii) _____
A) (i) $\Theta(\log n)$ (ii) $\Theta(n)$
B) (i) $\Theta(\log n)$ (ii) $\Theta(n \log n)$
C) (i) $\Theta(n)$ (ii) $\Theta(\log n)$
D) (i) $\Theta(n \log n)$ (ii) $\Theta(n)$
E) None of the above.

Question 6 [1 point] Le pire cas, en temps d'exécution, pour les tris suivants est:

	Tri par Insertion	Tri-fusion	Quicksort
A)	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$
B)	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n \log n)$
C)	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(n \log n)$
D)	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(n^2)$

E) Aucune de ces réponses.

Question 7 [1 point] Un arbre binaire est représenté avec un tableau:

I)

1	3	2	8	10	4	5	9

II)

10	70	80	110	90	-	-	120

III)

5	7	8	10	11	14	15	12

Lequel de ces tableaux correspond à un monceau-min (*min heap*)?
A) I seulement
B) I et II seulement
C) I et III seulement
D) II et III seulement
E) Aucun ne correspond à un monceau min.

Question 8 [1 point]
Soit le monceau max représenté à l'aide du tableau suivant:
`a[ ]=[10, 5, 3, 1, 4, 2].`
Quel sera l'état du tableau après une opération de retrait `removeMax()` ?
A) [5, 3, 1, 4, 2,-]
B) [2, 5, 3, 1, 4,-]
C) [5, 4, 3, 1, 2,-]
D) [5, 2, 3, 1, 4,-]
E) [5, 4, 3, 2, 1,-]

Question 9 [1 point]
Soit le monceau max représenté à l'aide du tableau suivant:
`a[ ]=[10, 5, 3, 1, 4, 2].`
Quel sera l'état du tableau après l'insertion de l'élément 9 dans le monceau?

A) [10,5,3,1,4,2,9]
B) [10,9,5,1,4,2,3]
C) [10,9,1,4,5,2,3]
D) [10,5,9,1,4,2,3]
E) Aucune de ces réponses

Question 10 [1 point] Soit la table de hachage ci-dessous pour laquelle les insertions sont effectuées avec la fonction de hachage suivante `h(k)= k mod 7`, et les collisions sont résolues avec le sondage quadratique.

0	1	2	3	4	5	6
	15	**1**	**8**		**5**	

Laquelle des affirmations suivantes est certainement correcte?

A) La clé 1 a été la dernière à être insérée.
B) La clé 8 a été la dernière à être insérée.
C) La clé 15 a été la dernière à être insérée.
D) Il est impossible de déterminer quelle clé a été insérée en dernier entre 8 et 5.
E) Aucune de ces affirmations n'est exacte.

Question 11 [1 point] Soit la table de hachage ci-dessous pour laquelle les insertions sont effectuées avec la fonction de hachage suivante `h(k)= k mod 7`, et les collisions sont résolues avec le sondage linéaire.

0	1	2	3	4	5	6
	15	**1**	**8**		**5**	

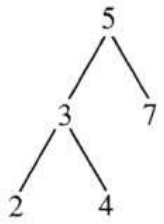
Le nombre moyen de sondage A pour trouver un élément se trouvant dans la table est :

A) A=1
B) $1 < A < 2$
C) A= 2
D) A >2
E) Aucune de ces réponses.

Question 12 [1 point] L'élément **2** est inséré dans la table de la Question 11 en utilisant encore le sondage linéaire. Puis, l'élément 3 est recherché dans la table. Combien de sondage seront requis (incluant le sondage d'une case vide) afin de s'apercevoir que cet élément ne se trouve pas dans la table?

A) 2 B) 3 C) 4 D) 5 E) 6 or more

Question 13 [1point] Un algorithme effectue la visite des noeuds de l'arbre ci-dessous dans l'ordre suivant: 2 4 3 7 5. Quel est cet algorithme?



A) Parcours pré-ordre.
B) Parcours post-ordre.
C) Recherche en profondeur à partir du nœud 5.
D) Recherche en largeur à partir du nœud 5.
E) Aucune de ces réponses.

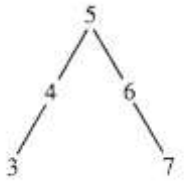
Question 14 [1 point] Soit un graphe pour lequel chacun des sommets a exactement 4 arêtes. Ce graphe est compose de N sommets et est représenté à l'aide d'une liste d'adjacence. Tenant compte du nombre fixe d'arêtes par sommet, la complexité asymptotique en temps d'exécution pour une recherche en profondeur (*depth-first search*) sera :

A) $\Theta(\log N)$
B) $\Theta(N)$
C) $\Theta(N \log N)$
D) $\Theta(N^2)$
E) Aucune de ces réponses.

Question 15 [1 point] Soit la procédure `stableSort(String [] A, int i)` permettant d'ordonner un tableau de mots de quatre lettres en se basant sur la lettre à la position i. Sachant que nous voulons un tri stable, lequel des algorithmes suivants va trier ces mots en ordre alphabétique?

A)
```
for (i=0 ; i<4 ; i++)  stableSort(A,i)
```
B)
```
for (i=3 ; i>=0 ; i--)  stableSort(A,i)
```
C)
```
for (i=0 ; i<4 ; i++)
      for (j=0 ; j<i ; j++)
         stableSort(A,i)
```
D)
```
for (i=0 ; i<=log n ; i++) stableSort(A,i)
```
E) Aucune de ces réponses.

Question 16 [2 points] Soit l'arbre AVL suivant:

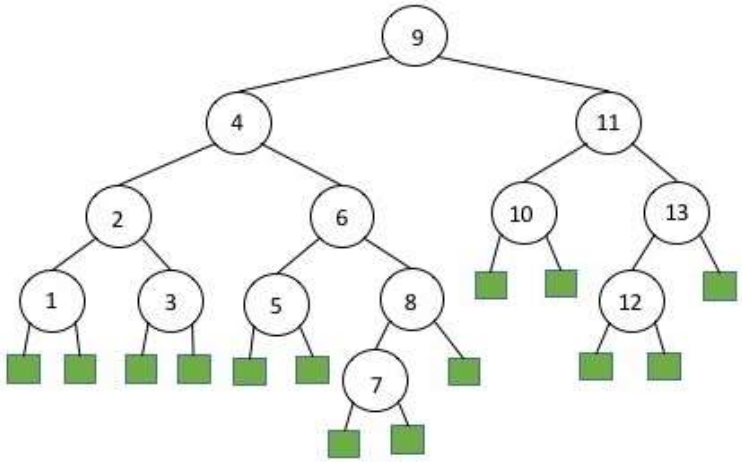


Montrer l'état de l'arbre après l'insertion des clés ci-dessous. Effectuer les opérations de ré-équilibrage lorsque requises:

Après insertion de 9	
Après insertion de 9 et 10	

Après insertion de 9, 10 ,11	
Après insertion de 9, 10, 11, 8	

Question 17 [2 points] Soit l'arbre AVL suivant:



Montrer les différentes étapes requises afin de retirer la clé 11 en déplaçant la clé 10 et en ré-équilibrant l'arbre AVL. Bien identifier les nœuds impliqués dans des éventuels ré-équilibrage (e.g. les nœuds X,Y,Z tels que vu en classe).

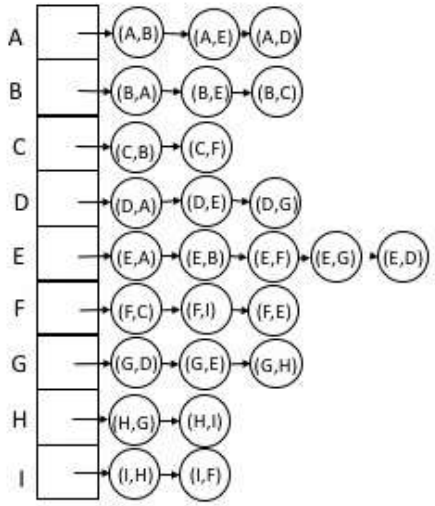
(page intentionally left blank)

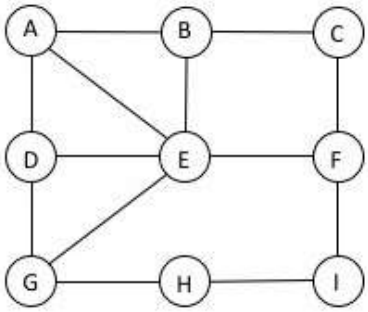
Question 18 [6 points=3+3] Soit le graphe ci-dessous et sa représentation en listes d'adjacence.

1) Prenant en compte l'ordre des arêtes spécifiées dans chacune des listes d'adjacence, effectuer une recherche en profondeur (*depth-first search*) à partir du sommet **A.**

a) Donner l'ordre de visite des sommets

b) Montrer sur le graphe l'arbre de recouvrement contenant les arêtes utilisées dans la découverte des sommets (*spanning tree of the DISCOVERY edges*).

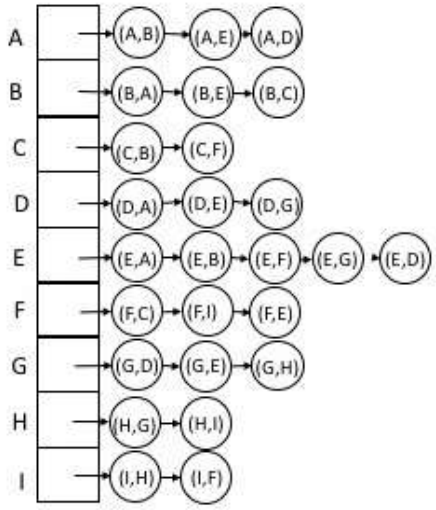


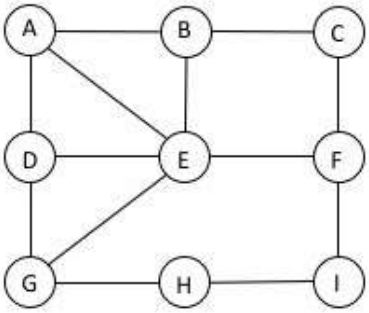


2) Prenant en compte l'ordre des arêtes spécifiées dans chacune des listes d'adjacence, effectuer une recherche en largeur (*breadth-first search*) à partir du sommet **A.**

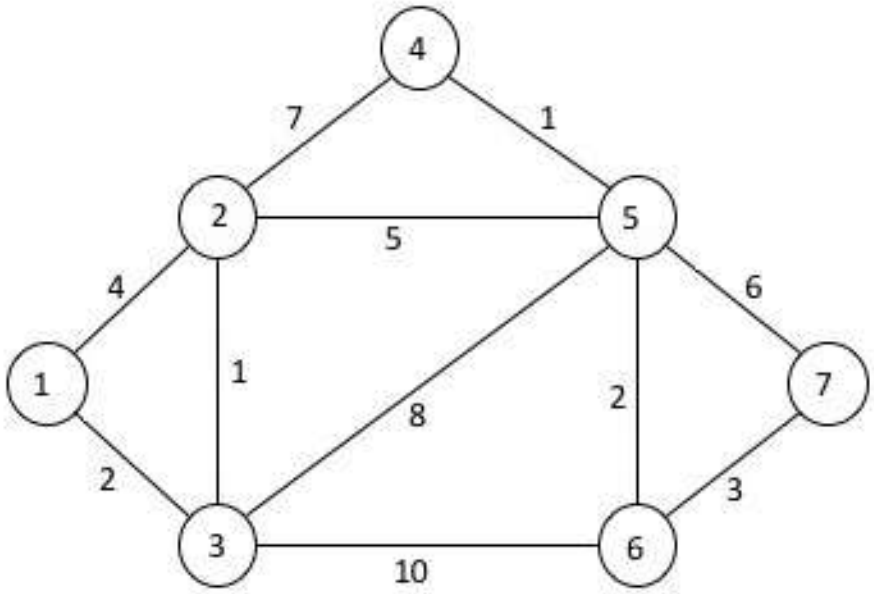
a) Donner l'ordre de visite des sommets

b) Montrer sur le graphe l'arbre de recouvrement contenant les arêtes utilisées dans la découverte des sommets (*spanning tree of the DISCOVERY edges*).





Question 19 [4 points]
Utiliser l'algorithme de Dijkstra afin d'obtenir l'arbre des plus courts chemins à partir du sommet **1** du graphe ci-dessous:



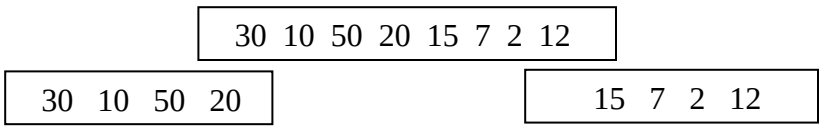
a) Donner la liste des sommets <u>dans l'ordre ou ceux-ci entre dans le nuage</u> de sommets produit par l'algorithme.
b) Donner la liste des arêtes <u>dans l'ordre ou celles-ci apparaissent dans l'arbre</u> des plus courts chemins. Utiliser le format suivant: (1,2) pour désigner l'arête entre les sommets 1 et 2.
c) Donner, pour chacun des sommets, la distance au sommet d'origine (sommet 1) tel que trouvée par l'algorithme.

Sommets

Arêtes

	dist
1	0
2	
3	
4	
5	
6	
7	

Question 20 [3 points]
Dessiner l'arbre de recursivité de l'algorithme de tri fusion (*mergesort*). Ici est montrée la première subdivision du tableau; vous devez montrer toutes les étapes de subdivision et de fusion jusqu'à ce que le tableau soit complètement trié.



Question 21 [2 points] Soit la fonction ci-dessous réalisant un tri-rapide (Quicksort) de façon récursive.

```
private static void doQuickSort(int[] data, int low, int high) {

        int i = low;
        int j = high;

        // beginning of Partition
        int pivot = data[low];

        while (i <= j) {
            while (data[i] < pivot) {
                i++;
            }
            while (data[j] > pivot) {
                j--;
            }
            if (i <= j) {
                swap(data, i, j);
                i++;
                j--;
            }
        }
        // *** end of Partition ***
        if (low < j)
            Sort.doQuickSort(data, low, j);
        if (i < high)
            Sort.doQuickSort(data, i, high);
    }
```

Une étape essentielle du tri rapide est celle ou la partition est effectuée. Cette partition permet de séparer le tableau en deux parties en utilisant un élément pivot. La valeur de ce pivot est déterminante pour l'efficacité de l'exécution de l'algorithme. Soit le tableau suivant:

18	35	12	20	15	7	2	50

a) Montrer l'état du tableau après l'exécution de la première partition (i.e. à la ligne indiqué dans le code source par `***`). L'appel initial est `doQuickSort(data,0,7);`

b) Changer la ligne `int pivot = data[low];` par la suivante:

```
int pivot = data[high];
```

Obtiendrez-vous le même résultat? Si non, dessiner l'état du tableau après partition

Question 22 [2 points]
Un programmeur écrit une fonction de recherche binaire dans un tableau, mais il commet une erreur. Il/Elle teste cette fonction et obtient le bon résultat mais la méthode semble plus lente que prévue.

a) Observer le code ci-dessous et évaluer la complexité, au pire cas, de l'algorithme tel que programmé.

   Cet algorithme s'exécute en O(_____)

b) Un bon algorithme de recherche binaire dans un tableau trié s'exécute en O(log N)
   Modifier le code ci-dessous afin qu'il réalise correctement la recherche binaire.
   Note: vous ne devez pas changer plus de 3 lignes de code.

```
// search if a given integer `value' is
// in a sorted array `data' of size N>0
  public int search(int value, int[] data, int N) {

    int inf=0, sup= N-1;
    int mid= (inf+sup)/2;

    while (data[mid]!=value && inf<sup) {

      if (data[mid]<value) {
        inf++;
      } else {
       sup--;
      }

      mid= (inf+sup)/2;
    }

    if (data[mid]==value)
      return mid;
    else
      return -1;
  }
```

Question 23 [1 points]
Soit un tableau contenant quatre éléments de valeur égale.

a) Combien de comparaisons seront effectuées si ce tableau est trié avec le tri par insertion?

b) Combien de comparaisons seront effectuées si ce tableau est trié avec le tri par sélection?

c) Combien d'opérations de fusion seront effectuées si ce tableau est trié avec le tri-fusion ?

Question 24 [3 points]
On vous donne un tableau A contenant des noms d'étudiants et leurs notes (chaque note est un entier entre 0 et 10). Écrire un algorithme permettant de trier ce tableau en ordre croissant de notes avec une complexité O(N), ou N est le nombre d'étudiants.
Vous pouvez utiliser un tableau auxiliaire afin de réaliser le tri.
Indice – utiliser le tri par paquets.
Rappel: Le tri par paquets (*BucketSort*) utilise les clés comme indices dans un tableau auxiliaire de séquences
Écrire votre algorithme en pseudo-cde ou en code Java. Vous pouvez utiliser les structures de données élémentaires vues en classe (tableaux, listes chainées).

```
public interface Student {
     int grade(); // key used for sorting
     String StudentName();
}
void BucketSort (Student[] A) {
```

(page intentionally left blank)