POLYTECHNIQUE MONTRÉAL

Questionnaire examen final

INF2010

Sigle du cours

Q1	
Q2	
Q3	
Q4	
Total	

Identification de l'étudiant(e)		
Nom : Kim	**Prénom : Victor**	
Signature :	**Matricule : 1954607**	**Groupe :**

Sigle et titre du cours		*Groupe*	*Trimestre*
INF2010 – Structures de données et algorithmes		**Tous**	**20201**
Professeur		***Local***	***Téléphone***
Tarek Ould Bachir		**Chez wam**	**2452**
Jour	***Date***	***Durée***	***Heures***
Lundi	**4 mai 2020**	**3h00**	**9h30-12h30**

Documentation	*Calculatrice*	
☒ Aucune	☐ Aucune	**Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.**
☐ Toute	☐ Toutes	
☒ Voir directives particulières	☒ Non programmable	

Directives particulières

Ne posez pas de question durant l'examen. En cas de doute sur le sens d'une question, énoncez clairement toute supposition que vous faites. Ne communiquez avec personne durant l'examen. Remettez un fichier ZIP incluant un PDF avec vos réponses et les codes sources demandés.

Important

Cet examen contient **5** questions sur un total de **16** pages **(excluant cette page)**

La pondération de cet examen est de **40** **%**

Vous devez répondre sur : ☒ le questionnaire ☐ le cahier ☐ les deux

Vous devez remettre le questionnaire : ☒ oui ☐ non

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Durée: 3h.
Pondération: 40%.
Documentation: Toute documentation permise.
Directives particulières:

- Répondre à toutes les questions, la valeur de chaque question est indiquée.
- Ne communiquez pas avec personne durant l'examen.
- Ne posez pas de question durant l'examen. En cas de doute sur le sens d'une question, énoncez clairement toute supposition que vous faites.
- Après avoir terminé l'examen, téléversez vers Moodle deux fichiers :
    - un fichier zip incluant les codes sources demandés; et,
    - un fichier PDF comportant votre nom, votre matricule, l'énoncé sur l'honneur et vos réponses.

Question 0. (0 points, mais obligatoire)

Dans votre fichier de réponse, ajoutez les informations et la mention suivantes :

Mon nom de famille est : Kim_____

Mon prénom est : Victor_____

Mon matricule est : 1954607_____

J'affirme, sur mon honneur, avoir fait cet examen tout/e seul/e sans l'aide d'une autre personne.

Réécrivez la phrase ci-après en accordant correctement le genre :

___ J'affirme, sur mon honneur, avoir fait cet examen tout/e seul/e sans l'aide d'une autre personne._____ ___.

Question 1 : Monceaux (5 points/20)

Considérez le code Java `Final20201BinaryHeap.java` qui vous est fourni dans le répertoire ./q1.

Vous devez être en mesure de compiler ce code après avoir complété la ligne 7 du fichier `Final20201BinaryHeap.java` avec vos informations personnelles :

```
private static final int MON_MATRICULE = _1954607_________; // <= A COMPLÉTER
```

La fonction principale (`main`) de ce code tente de caractériser la construction d'un monceau selon les deux approches discutées en classe : i) un élément à la fois; ii) Tous les éléments pris à la fois.

Trois jeux de données sont utilisés pour ce faire : `cle1`, `cle2` et `cle3`. Le code Java qui vous est fourni génère ces jeux de données au moyen, respectivement, des méthodes suivantes :

```
public static Integer[] cles1(int monMatricule, int nombreCles)
public static Integer[] cles2(int monMatricule, int nombreCles)
public static Integer[] cles3(int monMatricule, int nombreCles)
```

Trois tailles du jeu de données sont considérées à chaque fois : 1 000, 10 000 et 100 000.

a) **(1 point)** Complétez l'affichage résultant de l'exécution de la fonction principale `main` suite à l'ajout de votre identifiant personnel à la ligne 7 du fichier `Final20201BinaryHeap.java`. Pour ce faire, vous devez reproduire ci-après les nombre que produit votre version du code aux endroits identifiés par « `<= VALEUR À INDIQUER` »

```
### Nombre de clés: 1000
#Jeu de données: cles1
Nombre 1: _8987___________ <= VALEUR A INDIQUER
Nombre 2: _1492___________ <= VALEUR A INDIQUER
#Jeu de données: cles2
Nombre 1: _1000___________ <= VALEUR A INDIQUER
Nombre 2: _1000___________ <= VALEUR A INDIQUER
#Jeu de données: cles3
Nombre 1: _2044___________ <= VALEUR A INDIQUER
Nombre 2: _1422___________ <= VALEUR A INDIQUER

### Nombre de clés: 10000
#Jeu de données: cles1
Nombre 1: _123631___________ <= VALEUR A INDIQUER
Nombre 2: _14992___________ <= VALEUR A INDIQUER
```

```
#Jeu de données: cles2
Nombre 1: __10000__________ <= VALEUR A INDIQUER
Nombre 2: __10000__________ <= VALEUR A INDIQUER
#Jeu de données: cles3
Nombre 1: _20321___________ <= VALEUR A INDIQUER
Nombre 2: _14160___________ <= VALEUR A INDIQUER

### Nombre de clés: 100000
#Jeu de données: cles1
Nombre 1: __1568946__________ <= VALEUR A INDIQUER
Nombre 2: __149990__________ <= VALEUR A INDIQUER
#Jeu de données: cles2
Nombre 1: __100000__________ <= VALEUR A INDIQUER
Nombre 2: __100000__________ <= VALEUR A INDIQUER
#Jeu de données: cles3
Nombre 1: __202981__________ <= VALEUR A INDIQUER
Nombre 2: __141380__________ <= VALEUR A INDIQUER
```

Résumez vos réponses dans ce tableau :

		Taille du jeu de données		
		1 000	**10 000**	**100 000**
cles1	Nombre 1	8987	123631	1568946
	Nombre 2	1492	14992	149990
cles2	Nombre 1	1000	10000	100000
	Nombre 2	1000	10000	100000
cles3	Nombre 1	2044	20321	202981
	Nombre 2	1422	14160	141380

b) **(1 point)** Complétez le tableau suivant afin d'associer `Nombre 1` et `Nombre 2` à ce qu'ils mesurent. Pour ce faire, mettez un X dans les cases appropriées.

	Nombre total des percolations	
	Insertion un élément à la fois	**Insertion de tous les éléments à la fois**
`Nombre 1`	x	
`Nombre 2`		x

c) **(1 point)** Complétez le tableau suivant afin d'associer chacun des jeux de données à une des définitions proposées. Pour ce faire, mettez un X dans les cases appropriées.

	`cles1`	**`cles2`**	**`cles3`**
Données aléatoires			x
Données triées en ordre croissant		x	
Données partiellement triées en ordre croissant			
Données triées en ordre décroissant	x		
Données partiellement triées en ordre décroissant			

d) **(1 point)** Complétez le tableau suivant afin d'associer chacun des jeux de données à une des catégories proposées. Pour ce faire, mettez un X dans les cases appropriées.

	`cles1`	**`cles2`**	**`cles3`**
Meilleur cas		x	
Cas moyen			x
Pire cas	x		

e) **(1 point)** À la lumière de ce qui précède, discutez les valeurs obtenues pour `Nombre 1` lorsque le jeu de données `cle1` est considéré. Discutez comment ces valeurs se comparent aux autres jeux de données et comment ces valeurs croissent avec la taille du jeu de données.

Les valeurs du nombre 1 sont plus grand que les autres. Le nombre 1, sa clé 1 $\frac{123631}{8987} = 13.756 \ldots$ donc supérieur à 10 et les autres ont une croissance moins équivalent ou plus petit que 10.

Question 2 : Recherche de patron (5 points/20)

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. La chaîne de caractères recherchée est alors remplacée par un entier obtenu par un calcul de hachage. On considère dans cet exercice l'alphabet $\Sigma = \{$`A, B, … Z`$\}$, de sorte que nous aurons $d = |\Sigma| = 26$. On admettra l'encodage suivant :

Symbole	A	C	…	Z
Code	0	1	…	25

Le hachage est calculé en base d = 26. Par exemple, la séquence `"BAC"`, encodée **1, 0, 2** produira une valeur de hash :

$$(\mathbf{1}\times d + \mathbf{0})\times d + \mathbf{2} =$$
$$(\mathbf{1}\times 26 + \mathbf{0})\times 26 + \mathbf{2} =$$
$$678.$$

Une implémentation de cet algorithme vous est fournie dans le répertoire ./q2. Vous devriez être en mesure de compiler ce code après avoir complété les fonctions demandées ci-après, ainsi la ligne 9 du fichier `Final20201RabinKarp.java` où vous devez inscrire vos informations personnelles :

```
private static final int MON_MATRICULE = _1954607__________; // <= A COMPLÉTER
```

La fonction principale (`main`) décrite dans `Final20201RabinKarp.java` tente de retrouver dans un texte fourni un patron généré à partir de votre matricule étudiant. L'encodage des caractères vous est fourni par le membre statique `Map<Character, Integer> ENCODING` de la classe `Final20201RabinKarp`, et associe à chaque caractère 'A', 'B', …, 'Z' un entier, soit 0, 1, …, 25 respectivement. Cela se fait ainsi :

```
Integer value = ENCODING.get( charKey );
```

a) **(1 point)** Donnez le hash p produit par le patron P[1:5] = `"DERMK"` (ce patron s'obtient en utilisant le matricule 1625144). Justifiez votre réponse par un calcul.

Patron=EGPBV

E=4, G=6, P=15, B=1, V=21,

d=26

(((4*d)+6)*d+15)*d+1)*d+21)

=1943547

b) **(1 point)** Complétez la fonction `computeHash(String substring)` donnée dans le fichier `Final20201RabinKarp.java`. **Remettez le fichier modifié dans un zip avec votre examen.**

Aide : Assurez-vous que vous parvenez à retrouver le résultat de la question 2.a avec votre implémentation.

c) **(1 point)** Complétez la fonction `updateHash(int oldHash, char oldChar, char newChar)` qui implémente la fonction de récursion de Rabin Karp : $t_{s+1} = (t_s - hT[s+1])d + T[s+m+1]$, où $h = d^{m-1}$. **Remettez le fichier modifié dans un zip avec votre examen.**

d) **(1 point)** Reproduisez l'affichage résultat de l'exécution de la fonction principale `main` suite à la complétion des questions 2.b et 2.c et l'ajout de votre identifiant personnel à la ligne 9 du fichier `Final20201RabinKarp.java`.

Aide : Si votre matricule était 1625144, vous verriez

```
Patron recherche:DEMRK
Au décalage: 1520, on retrouve: DEMRK
```

Ce qui est cohérent avec le texte fourni aux lignes 26-52 :

```
26   private static final String text =
27         "RKAACGKFHGOHBBJCMAKJIWDBECJBBJJABJGADCDOMEGAKFOECALKEDKQEACG" +
28         "NQAEQLNBAOGTFCDPJAFOFQGFHJABAEIEGKBKGGAHDBCGJFEGJMSQAHOMJEII" +
…
51         "MEEJESODIJOGEGPBVEBAHQEBEBFBFLNNGKJLBAGICJBGLNEDCBEIDCHORECA" +
52          "LKCGDIMEHEBMAEQLNDEMDEMRK";
```

```
Patron Recherche: EGPBV
Au decalage:272  on retrouve :EGPBV
Au decalage:732  on retrouve :EGPBV
Au decalage:1452  on retrouve :EGPBV
```

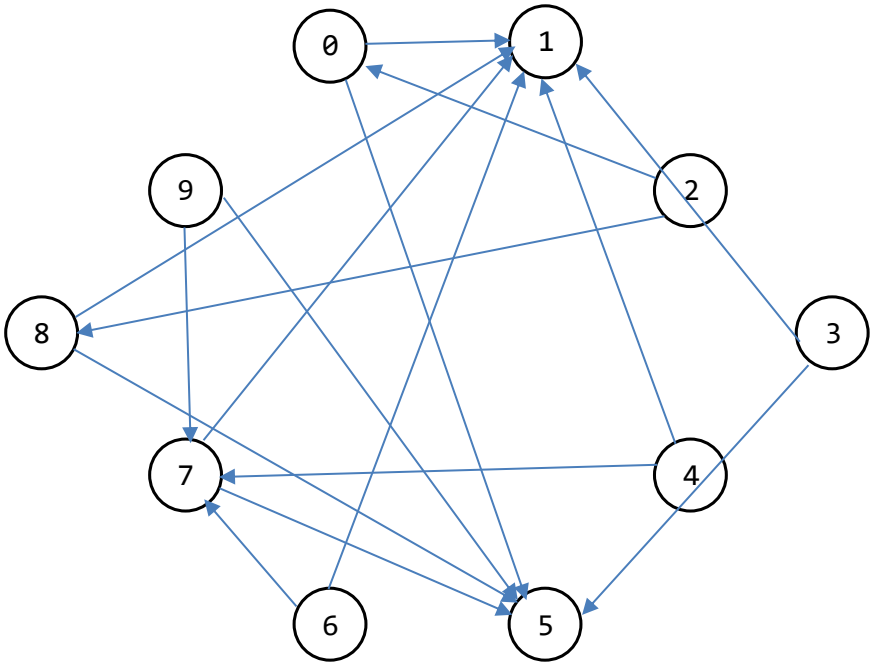
e) **(1 point)** Proposez une analyse coût/bénéfice de l'utilisation de la seconde formulation de Rabin Karp pour ce problème (calcul modulaire).

Question 3 : Ordre topologique (5 points/20)

On désire connaître l'ordre topologique d'un graphe dirigé acyclique. Une implémentation de cet algorithme vous est fournie dans le répertoire ./q3. Vous devriez être en mesure de compiler ce code après avoir complété la ligne 5 du fichier q3.java :

```
private static final int MON_MATRICULE =1954607 ; // <= A COMPLÉTER
```

a) **(1 point)** Reproduisez ci-après le graphe dirigé acyclique généré par q3.java une fois que votre identifiant y est inscrit à la ligne 5 du fichier q3.java. **Remettez le fichier modifié dans un zip avec votre examen.**



La classe `TopologicalOrder` est censée déterminer l'ordre topologique du graphe dirigé acyclique selon les deux méthodes vues en cours, soit : i) Celle utilisant une file comme liste de travail; ii) celle utilisant un parcours DSF post-ordre inverse.

b) **(1 point)** Complétez la méthode `findTopologicalOrder1()` de la classe `TopologicalOrder` qui implémente l'algorithme utilisant une liste de travail vu en classe. Confirmez le bon fonctionnement de votre implémentation en donnant l'ordre retourné par votre implémentation. Débutez la numérotation à 1. Vous pouvez vous inspirer du pseudocode vu en classe. **Remettez le fichier modifié dans un zip avec votre examen.**

Ordre trouvé (la numérotation débute à 1) :

Nœud	0	1	2	3	4	5	6	7	8	9
Ordre :	6	9	1	2	3	10	4	8	7	5

Aide : Si votre programme générait l'affichage suivant :

```
DAG:
10
16
0->4
0->7
2->6
2->9
3->4
3->6
5->0
5->1
6->1
6->4
7->1
7->4
8->1
8->4
9->7
9->8

Topological Order 1
0 6
1 9
2 1
3 2
4 10
5 3
6 5
7 8
8 7
9 4
…
```

Alors l'ordre trouvé serait (la numérotation débute à 1) :

Nœud	0	1	2	3	4	5	6	7	8	9
Ordre :	6	9	1	2	10	3	5	8	7	4

c) **(1 point)** Confirmez le résultat produit par votre implémentation en complétant la table ci-après :

Nœud\Itération	1	2	3	4	5	6	7	8	9	10
0	1	0	-	-	-	-	-	-	-	-
1	6	6	5	4	3	3	2	1	0	-
2	0	-	-	-	-	-	-	-	-	-
3	0	-	-	-	-	-	-	-	-	-
4	0	-	-	-	-	-	-	-	-	-
5	5	5	4	4	4	3	2	1	0	-
6	0	-	-	-	-	-	-	-	-	-
7	3	3	3	2	1	0	-	-	-	-
8	1	0	-	-	-	-	-	-	-	-
9	0	-	-	-	-	-	-	-	-	-
Nœuds entrant	2,3,4,6,9	4,6,9,0,8	6,9,0,8	9,0,8	0,8	8,7	7	-	1,5	-
Nœud sortant	2	3	4	6	9	0	8	7	1	5

d) **(1 point)** Complétez la méthode `findTopologicalOrder2()` de la classe `TopologicalOrder` qui implémente l'algorithme utilisant le parcours DFS post-ordre inverse. Confirmez le bon fonctionnement de votre implémentation en donnant l'ordre retourné par votre implémentation. Débutez la numérotation à 1. Vous pouvez vous inspirer du code Java vu en classe. **Remettez le fichier modifié dans un zip avec votre examen.**

Ordre trouvé (la numérotation débute à 1) :

Nœud	0	1	2	3	4	5	6	7	8	9
Ordre :	8	10	6	5	3	9	2	4	7	1

Aide : Si votre programme générait l'affichage suivant :

```
DAG:
10
16
…
Topological Order 2
0 7
1 9
2 3
3 2
4 10
5 1
6 6
7 8
8 5
9 4

…
```

Alors l'ordre trouvé serait (la numérotation débute à 1) :

Nœud	0	1	2	3	4	5	6	7	8	9
Ordre :	7	9	3	2	10	1	6	8	5	4

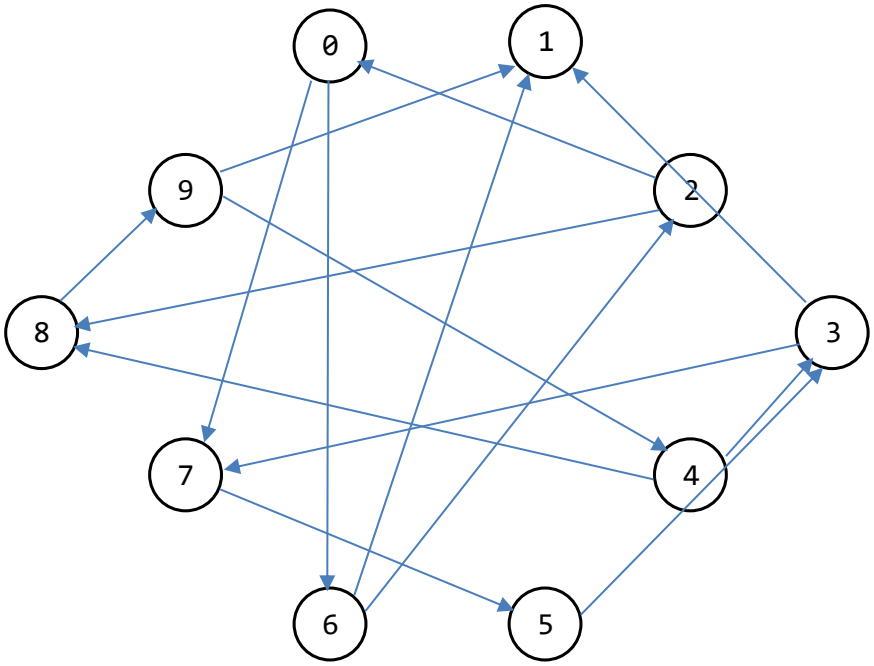
e) **(1 point)** Reproduisez l'affichage résultat de l'exécution de la fonction principale `main` du suite à la complétion des questions 3.b et 3.d et l'ajout de votre identifiant personnel à la ligne 5 du fichier `q3.java`.

Question 4 : Composantes fortement connexes (2.5 points/20)

On désire connaître les composantes fortement connexes d'un graphe dirigé. Une implémentation de cet algorithme vous est fournie dans le répertoire ./q4. Vous devriez être en mesure de compiler ce code après avoir complété la ligne 5 du fichier q4.java :

```
private static final int MON_MATRICULE =1954607 ; // <= A COMPLÉTER
```

a) **(0.5 point)** Reproduisez ci-après le graphe dirigé acyclique généré par q4.java une fois que votre identifiant y est inscrit à la ligne 5 du fichier q4.java. **Remettez le fichier modifié dans un zip avec votre examen.**



La classe `CFC` est censée déterminer les composantes fortement connexes du graphe dirigé selon la méthode de Kosaraju vue en cours. L'algorithme procède en exécutant deux DSF successifs. Le premier s'exécute sur le graphe transposé G^T et détermine l'ordre de visite des nœuds lors du d'un second DSF opéré sur le graphe d'origine.

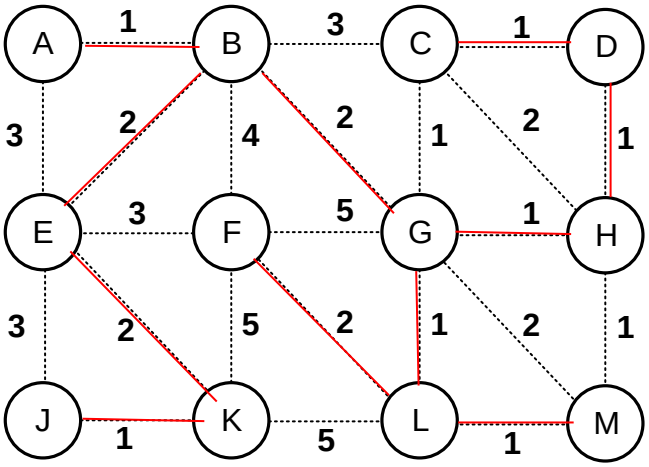
b) **(1 point)** Complétez la méthode `findOrder(…)`de la classe `CFC` qui déterminer l'ordre de visite des nœuds durant le second DFS. Pour ce faire, il vous faudra compléter la méthode `dfs1(…)` également. **Remettez le fichier modifié dans un zip avec votre examen.**

c) **(1 point)** Reproduisez l'ordre retourné par votre implémentation en complétant le tableau suivant, où chacun des nœuds `0` à `9` est associé à une composante fortement connexe que vous implémentation de Kosaraju aura identifiée. Chaque colonne représente une composante. Un nœud X est associé à une composante Y en noircissant la case correspondant à l'intersection de la ligne X et de la colonne Y. La numérotation des composantes débute à 1.

Nœud	Composante						
	1	2	3	4	5	6	7
0	x						
1							
2	x						
3		x					
4			x				
5		x					
6	x						
7		x					
8			x				
9			x				

Question 5 : Arbre sous-tendant minimum (2.5 points/20)

Donnez l'arbre sous-tendant minimum obtenu par l'algorithme de Kruskal en noircissant les arêtes retenues dans le graphe ci-après. Donnez le coût de l'arbre ainsi obtenu.



Coût: __15_____

Vous pouvez vous aider du tableau donné à la page suivante. Le tableau n'est pas noté.

Kruskal :

Arête	Coût	Retenue?
(A, B)	1	x
(A, E)	3	
(B, C)	3	
(B, E)	2	x
(B, F)	4	
(B, G)	2	x
(C, D)	1	x
(C, G)	1	
(C, H)	2	
(D, H)	1	x
(E, F)	3	
(E, J)	3	
(E, K)	2	x
(F, G)	5	
(F, K)	5	
(F, L)	2	x
(G, H)	1	x
(G, L)	1	x
(G, M)	2	
(H, M)	1	
(J, K)	1	x
(K, L)	5	
(L, M)	1	x