POLYTECHNIQUE MONTRÉAL

Questionnaire examen final

INF2010

Sigle du cours

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

Sigle et titre du cours	*Groupe*	*Trimestre*
INF2010 – Structures de données et algorithmes	**Tous**	**20141**
Professeur	***Local***	***Téléphone***
Ettore Merlo, responsable / Tarek Ould Bachir, chargé	**M-5028**	**7128 / 5193**

Jour	*Date*	*Durée*	*Heures*
Samedi	**19 avril 2014**	**2h30**	**13h30-16h00**

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

Directives particulières

💻 Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.

Important

Cet examen contient **6** questions sur un total de **21** 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.

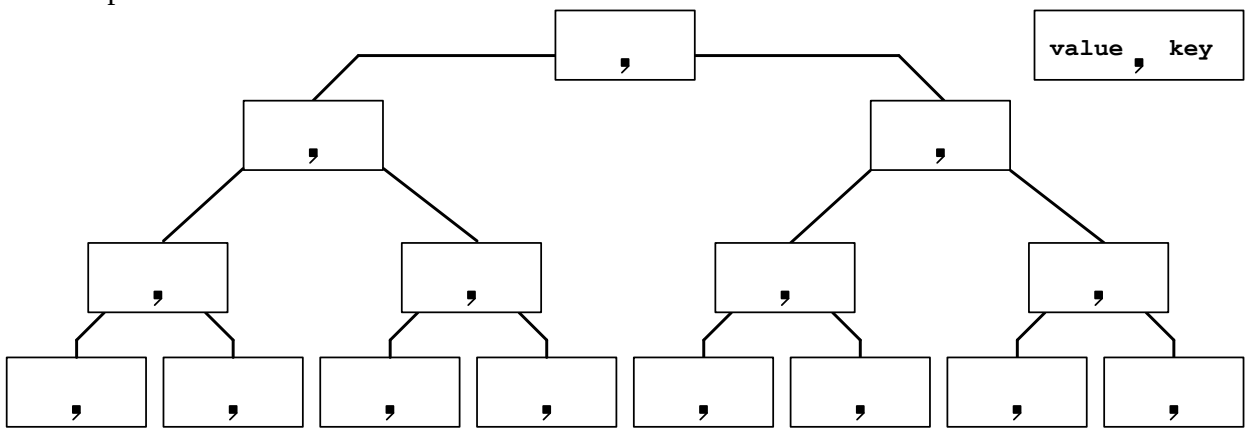
Question 1 : Monceaux (20 points)

Pour cette question, vous pouvez vous référez au code Java de l'Annexe 1.

a) (2 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après. Indiquez dans les cases les valeurs de **`value`** et **`key`** de l'objet **`Entry`** contenues dans le monceau.

Indice	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Contenu	-	14	21	18	36	35	27	42	70	45	63	90	63	81	49	56

Votre réponse :



b) Considérez la fonction `changeKey(int location, int newKey)` donnée à l'Annexe 1 permettant de modifier la clé d'une entrée du monceau.

b.1) (2 pts) Quelle est la complexité asymptotique de cette fonction en pire cas ? Justifiez clairement votre réponse. Une réponse non justifiée ne sera pas considérée.

b.2) (2 pts) Quelle est la complexité asymptotique de cette fonction en meilleur cas ? Justifiez clairement votre réponse. Une réponse non justifiée ne sera pas considérée.

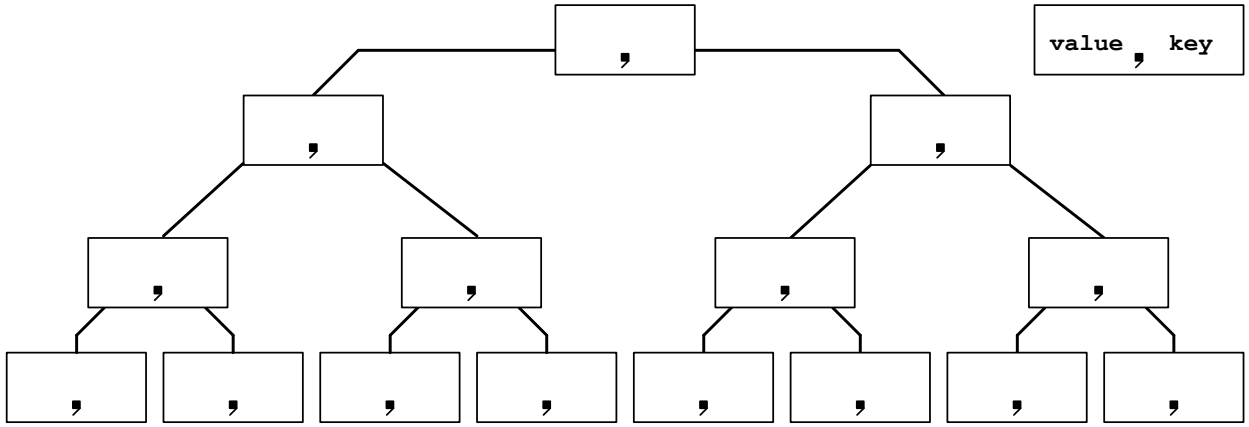
c) (3 pts) En vous fiant au code donné à l'Annexe 1, dessinez le monceau résultant de l'appel : **`BinaryHeap(values_1, true)`**

Indiquez dans les cases les valeurs de **`value`** et **`key`** de l'objet **`Entry`** contenues dans le monceau.

Le tableau **`values_1`**est le suivant:

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenu	36	18	45	81	63	27	90	70	21	35	42	63	14	49	56

Monceau résultant :



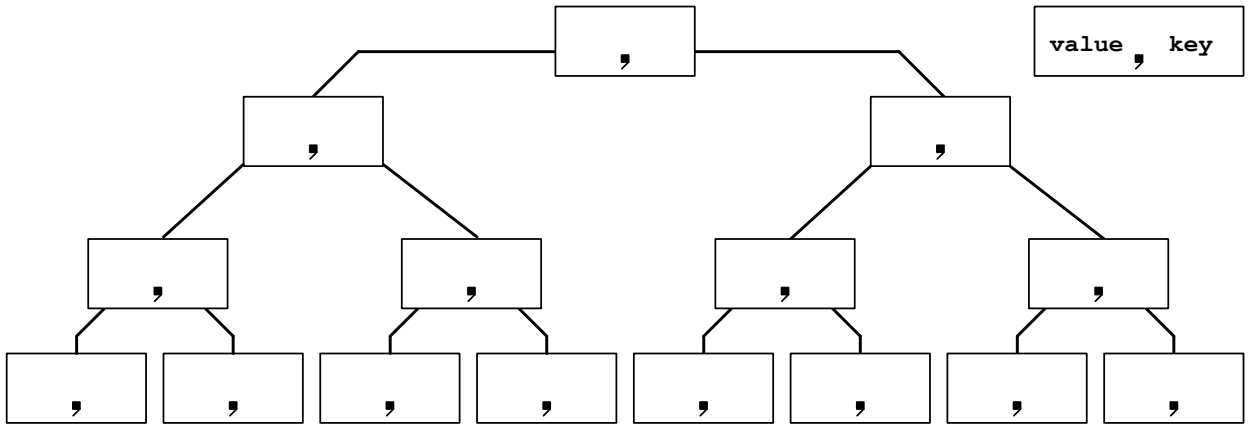
d) (3 pts) En vous fiant au code donné à l'Annexe 1, dessinez le monceau résultant de l'appel : **`BinaryHeap(values_1, false)`**

Indiquez dans les cases les valeurs de **`value`** et **`key`** de l'objet **`Entry`** contenues dans le monceau.

Le tableau **`values_1`** est le suivant:

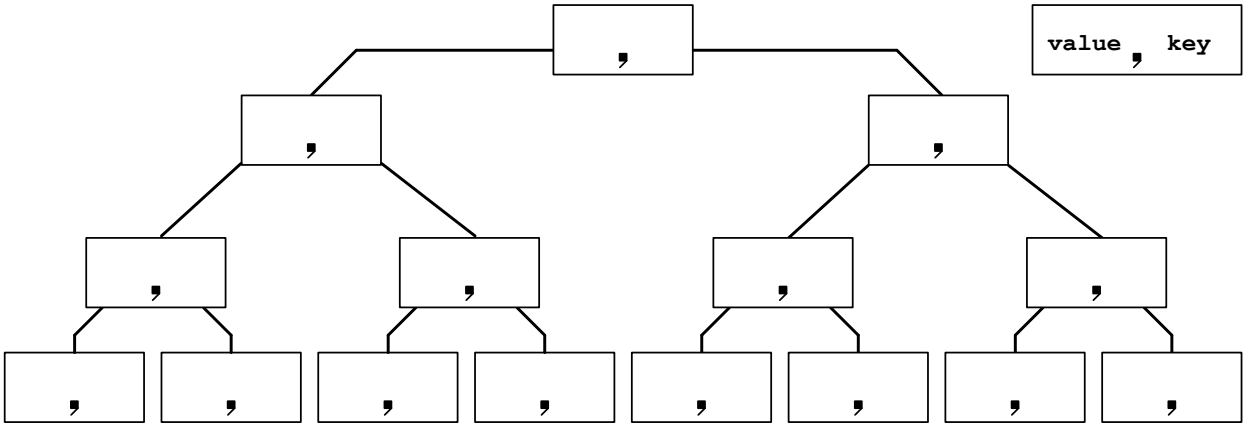
Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenu	36	18	45	81	63	27	90	70	21	35	42	63	14	49	56

Monceau résultant :

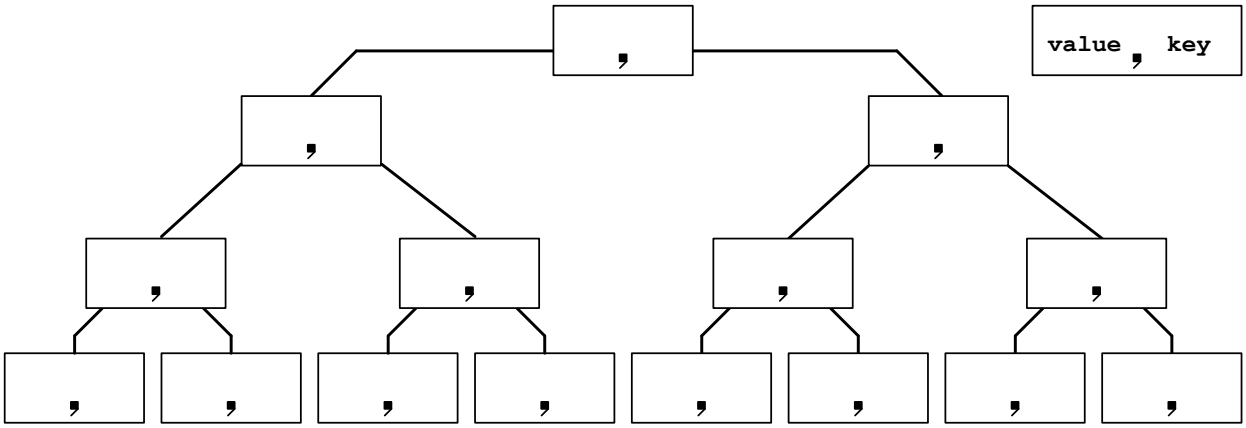


e) Dessinez l'état du monceau de la question 1.c) suite à deux appels consécutifs à `deleteRoot()` :

e.1) (1 pt) Monceau résultant du premier `deleteRoot()`. Indiquez dans les cases les valeurs de **`value`** et **`key`** de l'objet **`Entry`** contenues dans le monceau.

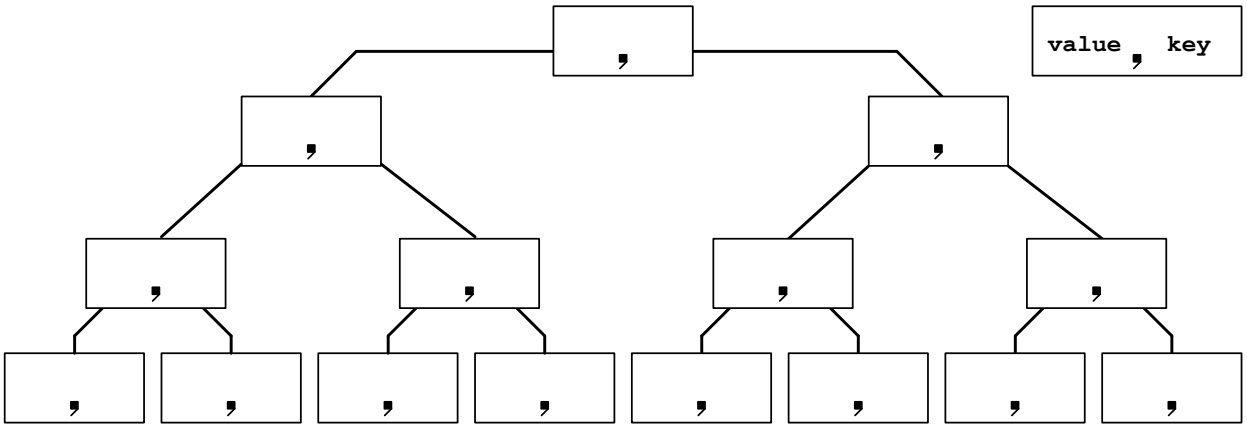


e.2) (1 pt) Monceau résultant du second `deleteRoot()`. Indiquez dans les cases les valeurs de **`value`** et **`key`** de l'objet **`Entry`** contenues dans le monceau.

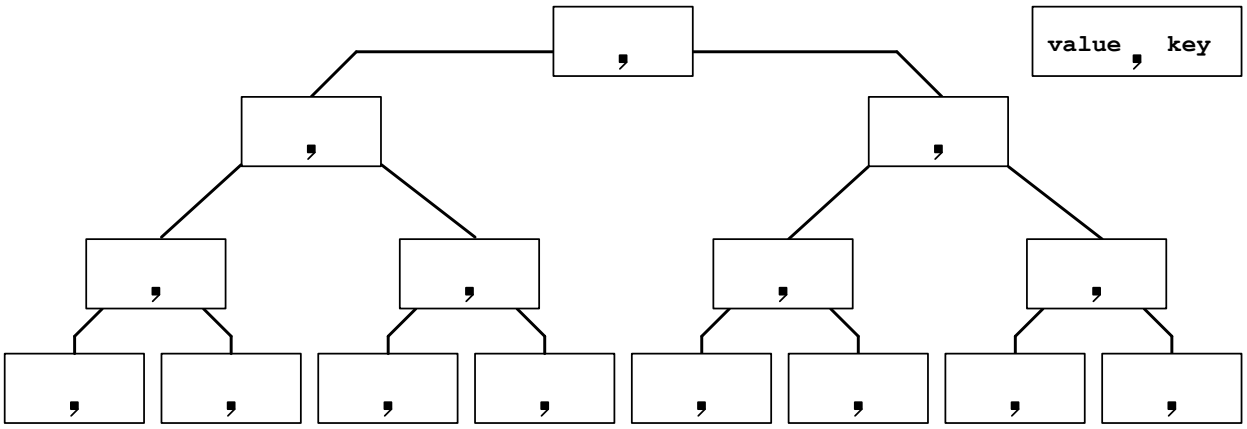


f) Dessinez l'état du monceau de la question 1.d) suite à deux appels consécutifs à `deleteRoot()` :

f.1) (1 pt) Monceau résultant du premier `deleteRoot()`. Indiquez dans les cases les valeurs de **`value`** et **`key`** de l'objet **`Entry`** contenues dans le monceau.



f.2) (1 pt) Monceau résultant du second `deleteRoot()`. Indiquez dans les cases les valeurs de **`value`** et **`key`** de l'objet **`Entry`** contenues dans le monceau.

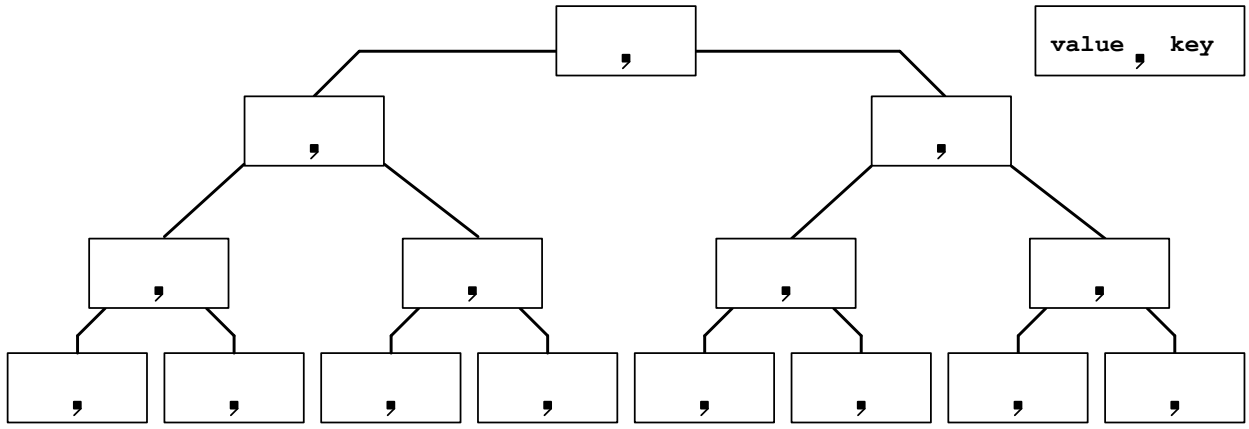


g) (4 pts) Exécutez `changeKey(15, 15)` sur le monceau suivant construit résultant de l'appel : **`BinaryHeap(values_2)`**. Indiquez dans les cases les valeurs de **`value`** et **`key`** de l'objet **`Entry`** contenues dans le monceau.

Le tableau **`values_2`** est le suivant:

Indice	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Contenu	15	16	15	30	29	15	16	31	30	30	39	16	15	16	17

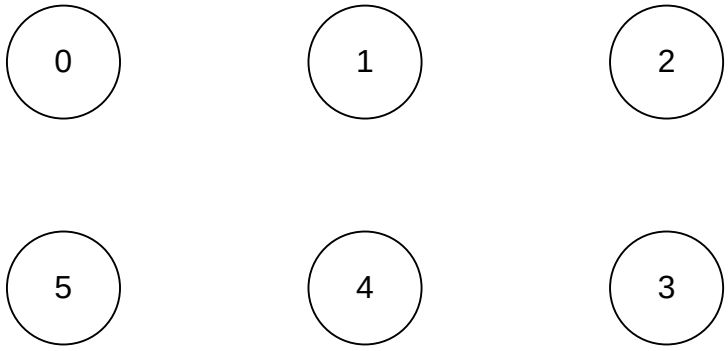
Résultat:



Question 2 : Recherche de patron (14 points)

On vous demande de retrouver le patron P[1:8] = « eenne » dans un texte.

a) (4 pnts) Dessiner le diagramme d'états de l'automate à états finis permettant de ce faire :



b) (4 pnts) Donner la table de transitions de l'automate recherché.

q\a	e	n	Autre
0			
1			
2			
3			
4			
5			

c) (6 pts) Quels seront respectivement le ou les états les plus visités et le ou les états les moins visités par l'automate une fois arrivé à la fin tu texte suivante.

T[1 : 34] = « eeeeeeeennneneeeeenneeeenneeenneee »

Aidez-vous de la table suivante:

T[i]	**e**	**e**	**e**	**e**	**e**	**e**	**e**
q							
T[i]	**e**	**n**	**n**	**n**	**e**	**n**	**e**
q							
T[i]	**e**	**e**	**e**	**e**	**n**	**n**	**e**
q							
T[i]	**e**	**e**	**e**	**n**	**n**	**e**	**e**
q							
T[i]	**e**	**n**	**n**	**e**	**e**	**e**	
q							

Le ou les états les plus visités : _____

Le ou les états les moins visités : _____

Question 3 : Programmation dynamique (15 points)

a) (10 points) En utilisant l'algorithme vu en classe, donnez la longueur de la PLSC des séquences de caractères X = "abcbcaacbabc" et Y = "bbacbabcbcabca".

Aidez-vous du tableau donné ci-après. Inscrivez votre réponse à la page suivante.

X\Y		**a**	**b**	**c**	**b**	**c**	**a**	**a**	**c**	**b**	**a**	**b**	**c**
	0	0	0	0	0	0	0	0	0	0	0	0	0
b	0												
b	0												
a	0												
c	0												
b	0												
a	0												
b	0												
c	0												
b	0												
c	0												
a	0												
b	0												
c	0												
a	0												

Longueur de la PLSC : _____

PLCS trouvée : _____

b) (5 pts) Il existe deux autres PLSC pour les séquences X et Y que celle trouvée à la question 3.a. Donnez-les:

2^{e} PLCS : _____

3^{e} PLCS : _____

Question 4 : Programmation dynamique (20 points)

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$A_1 : 2 \times 1$; $A_2 : 1 \times 1$; $A_3 : 1 \times 3$; $A_4 : 3 \times 1$; $A_5 : 1 \times 6$

Considérez les tables **m** et **s** obtenue par l'exécution de l'algorithme dynamique vu en cours.

m	**1**	**2**	**3**	**4**	**5**
1	**0**	**2**			
2		**0**	**3**	**10**	**16**
3			**0**	**3**	**15**
4				**0**	**18**
5					**0**

s	**1**	**2**	**3**	**4**	**5**
1		**1**			
2			**2**	**2**	**4**
3				**3**	**4**
4					**4**
5					

Compléter cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

a) (3 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_3. Donnez son coût.

Parenthésage optimal: A_1 A_2 A_3

Coût: _____

b) (4 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_4. Donnez son coût.

Parenthésage optimal: A_1 A_2 A_3 A_4

Coût: _____

c) (5 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_5. Donnez son coût.

Parenthésage optimal: A_1 A_2 A_3 A_4 A_5

Coût: _____

d) (8 pts) Si A_1, A_2 et A_3 étaient des matrices 2×2, 2×3 et 3×3 respectivement, quel serait le parenthésage optimal pour multiplier A_1 à A_4 ? Quel serait son coût ? Aidez-vous des matrices suivantes pour répondre à la question.

m	**1**	**2**	**3**	**4**	**5**
1	**0**				
2		**0**			
3			**0**		
4				**0**	
5					**0**

s	**1**	**2**	**3**	**4**	**5**
1		**1**			
2			**2**		
3				**3**	
4					**4**
5					

Parenthésage optimal: A_1 A_2 A_3 A_4

Coût: _____

Question 5 : Ordre topologique (13 points)

a) (7 pts) Donnez l'ordre topologique du graphe suivant en appliquant l'algorithme utilisant une file vu en classe.

$$V = \{A, B, C, D, E, F, G\}$$

$$E = \{(A, D), (C, F), (D, F), (D, G), (E, B), (E, C)\}$$

Nœud	1	2	3	4	5	6	7
A							
B							
C							
D							
E							
F							
G							
Entrée							
Sortie							

Ordre trouvé (débuter la numérotation à 1) :

Nœud	A	B	C	D	E	F	G
Ordre :							

b) (6 pts) Ce graphe admet au moins trois autres ordres topologiques. Donnez-les. D'autres tables vous sont fournies à l'Annexe 2. Utilisez-les si nécessaire.

Ordre alternatif #1 :

Nœud	A	B	C	D	E	F	G
Ordre:							

Ordre alternatif #2 :

Nœud	A	B	C	D	E	F	G
Ordre:							

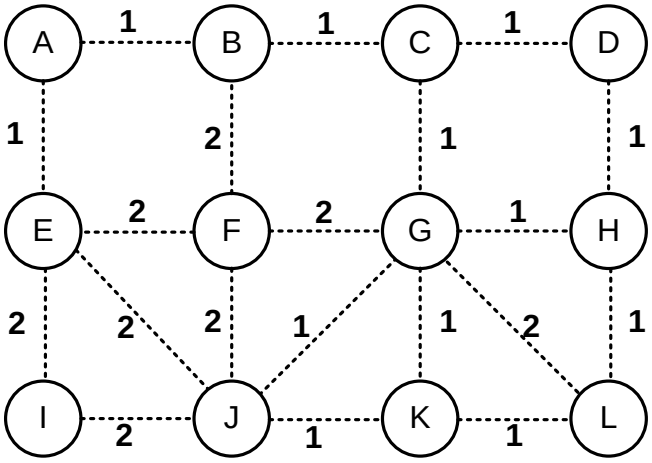
Ordre alternatif #3 :

Nœud	A	B	C	D	E	F	G
Ordre:							

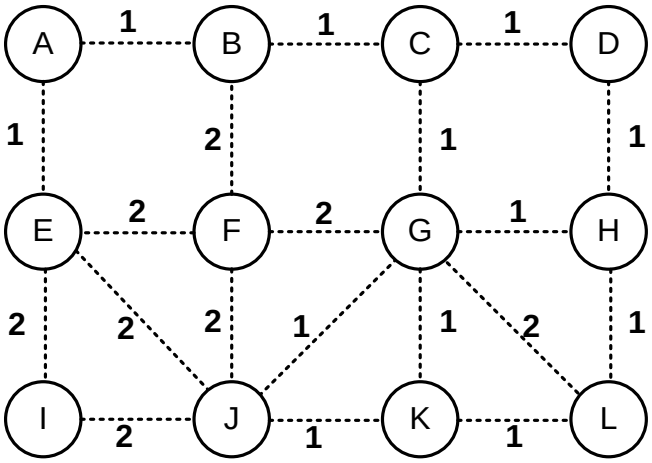
Question 6 : Arbre sous-tendant minimum (18 points)

Donnez les arbres sous-tendant minimum obtenus par les algorithmes de Boruvka, Prim (le noeud de départ étant A) et de Kruskal. Respectez l'ordre alphabétique pour effectuer vos traitements (pour visiter les composantes, les nœuds voisins ou les arêtes). Utilisez les tables fournies pour ce faire (le remplissage des tables compte dans l'attribution des points pour la question, sauf dans Boruvka ou seule la réponse finale compte).

Par Boruvka:



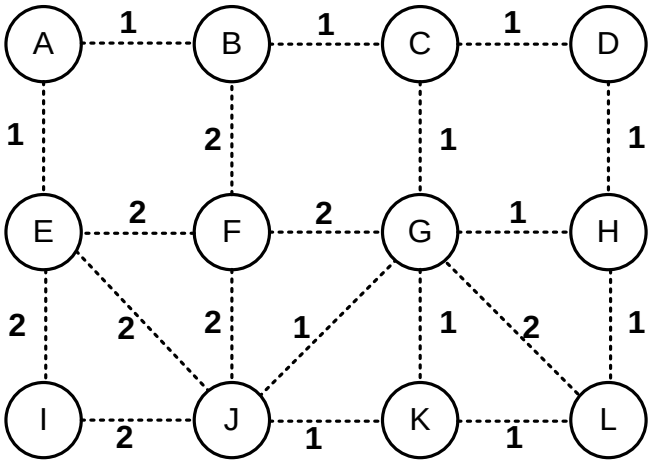
Par Prim:



Prim:

Nœud	Distance	Parent	Connu?
A			
B			
C			
D			
E			
F			
G			
H			
I			
J			
K			
L			

Par Kruskal:



Kruskal:

Arête	Poids	Retenue?
(A,B)	1	
(A,E)	1	
(B, C)	1	
(B, F)	2	
(C, D)	1	
(C, G)	1	
(D, H)	1	
(E, F)	2	
(E, I)	2	
(E, J)	2	
(F, G)	2	
(F, J)	2	
(G, H)	1	
(G, J)	1	
(G, K)	1	
(G, L)	2	
(H, L)	1	
(I, J)	2	
(J, K)	1	
(K, L)	1	

Annexe 1

```
public class BinaryHeap<AnyType>
{
   public BinaryHeap( )
   {
      this( DEFAULT_CAPACITY );
   }

   @SuppressWarnings("unchecked")
   public BinaryHeap( int capacity )
   {
      currentSize = 0;
      isMinHeap = true;
      array = new Entry[ capacity + 1 ];
   }

   @SuppressWarnings("unchecked")
   public BinaryHeap( AnyType [ ] items )
   {
      currentSize = items.length;
      array = new Entry[ ( currentSize + 2 ) * 11 / 10 ];

      int i = 1;
      for( AnyType item : items )
         array[ i++ ] = new Entry<AnyType>(item);

      buildHeap( );
   }

   @SuppressWarnings("unchecked")
   public BinaryHeap( AnyType [ ] items, boolean isMinHeap )
   {
      this.isMinHeap = isMinHeap;
        currentSize = items.length;
      array = new Entry[ ( currentSize + 2 ) * 11 / 10 ];

      int i = 1;
      for( AnyType item : items )
         array[ i++ ] = new Entry<AnyType>(item);

      buildHeap( );
   }

   public void insert( AnyType x )
   {
      if( currentSize == array.length - 1 )
         enlargeArray( array.length * 2 + 1 );

      // Percolate up
      int hole = ++currentSize;
      for( ; hole > 1 && x.hashCode() < array[ hole / 2 ].hashCode(); hole /= 2 )
         array[ hole ] = array[ hole / 2 ];
      array[ hole ] = new Entry<AnyType>(x);
   }
```

```
private void insert( Entry<AnyType> x )
{
   if( currentSize == array.length - 1 )
      enlargeArray( array.length * 2 + 1 );

   // Percolate up
   int hole = ++currentSize;
   for( ; hole > 1 && x.hashCode() < array[ hole / 2 ].hashCode(); hole /= 2 )
      array[ hole ] = array[ hole / 2 ];
   array[ hole ] = x;
}

@SuppressWarnings("unchecked")
private void enlargeArray( int newSize )
{
      if( newSize <= currentSize ) return;

      Entry<AnyType>[] old = array;
   array = new Entry[ newSize ];
   for( int i = 0; i < old.length; i++ )
      array[ i ] = old[ i ];
}

public AnyType deleteRoot( )
{
   if( isEmpty( ) ) return null;

   AnyType rootItem = array[ 1 ].value;
   array[ 1 ] = array[ currentSize-- ];
   percolateDown( 1 );

   return rootItem;
}

private void buildHeap( )
{
   for( int i = currentSize / 2; i > 0; i-- )
      percolateDown( i );
}

public boolean isEmpty( )
{
   return currentSize == 0;
}

public void makeEmpty( )
{
   currentSize = 0;
   array = null;
}
```

```
private void percolateDown( int hole )
{
   int child;
   Entry<AnyType> tmp = array[ hole ];

   for( ; hole * 2 <= currentSize; hole = child )
   {
      child = hole * 2;
      if( child != currentSize &&
           ( ((array[ child + 1 ].key < array[ child ].key) && isMinHeap) ||
             ((array[ child + 1 ].key > array[ child ].key) && !isMinHeap) ) )
         child++;
      if( array[ child ].key < tmp.key && isMinHeap ||
           array[ child ].key > tmp.key && !isMinHeap )
         array[ hole ] = array[ child ];
      else
         break;
   }
   array[ hole ] = tmp;
}

public String printFancyTree( )
{
   return printFancyTree( 1, "");
}

private String printFancyTree(int index, String prefix)
{
   String outputString = "";

   outputString = prefix + "|__";

   if( index <= currentSize )
   {
      boolean isLeaf = index > currentSize/2;

      outputString += array[ index ] + "\n";

      String _prefix = prefix;

      if( index%2 == 0 )
         _prefix += "|  "; // un | et trois espace
      else
         _prefix += "   " ; // quatre espaces

      if( !isLeaf )
      {
         outputString += printFancyTree( 2*index, _prefix);
         outputString += printFancyTree( 2*index + 1, _prefix);
      }
   }
   else
      outputString += "null\n";

   return outputString;
}
```

```
public void changeKey(int location, int newKey)
{
   if( location < 1 ) return;
   if( location > currentSize ) return;

   Entry<AnyType> tmp = array[location];
   tmp.setKey( newKey );

   array[location] = array[currentSize--];

   percolateDown( location );

   insert( tmp );
}

private static final int DEFAULT_CAPACITY = 10;

private boolean isMinHeap = true;
private int currentSize;
private Entry<AnyType> [] array;

private static class Entry<AnyType>
{
   public int key;
   public AnyType value;

   public Entry(AnyType value)
   {
      this.key = value.hashCode();
      this.value = value;
   }

   public Entry(int key, AnyType value)
   {
      this.key = key;
      this.value = value;
   }

   public void setKey(int key)
   {
      this.key = key;
   }

   public boolean equals(Object cmp)
   {
      return this.hashCode() == cmp.hashCode();
   }

   public int hashCode()
   {
      return key;
   }
```

```
   public String toString()
      {
         return "(v-" + value.toString() + ", k-" + key + ")";
      }
   }

   public static void main(String[] args)
   {
      Integer[] values_1 = {36, 18, 45, 81, 63, 27, 90, 70,
                            21, 35, 42, 63, 14, 49, 56};

         BinaryHeap<Integer> heap = new BinaryHeap<Integer>(values_1, true);

         System.out.println("Monceau Q1.c");
         System.out.println( heap.printFancyTree() );

         heap = new BinaryHeap<Integer>(values_1, false);

         System.out.println("Monceau Q1.d");
         System.out.println( heap.printFancyTree() );


         heap = new BinaryHeap<Integer>(values_1, true);

         heap.deleteRoot();
         System.out.println("Monceau Q1.e.1");
         System.out.println( heap.printFancyTree() );

         heap.deleteRoot();
         System.out.println("Monceau Q1.e.2");
         System.out.println( heap.printFancyTree() );


         heap = new BinaryHeap<Integer>(values_1, false);

         heap.deleteRoot();
         System.out.println("Monceau Q1.f.1");
         System.out.println( heap.printFancyTree() );

         heap.deleteRoot();
         System.out.println("Monceau Q1.f.2");
         System.out.println( heap.printFancyTree() );

         Integer[] values_2 = {15, 16, 15, 30, 29, 15, 16, 31,
                               30, 30, 39, 16, 15, 16, 17};
         heap = new BinaryHeap<Integer>(values_2);

        System.out.println("Monceau Q1.g");
        System.out.println( heap.printFancyTree() );
        System.out.println("Modify Key");
        int location = 15;
        int newKey = 15;
        heap.changeKey(location, newKey);
        System.out.println( heap.printFancyTree() );
   }
}
```

Annexe 2

Nœud	1	2	3	4	5	6	7
A							
B							
C							
D							
E							
F							
G							
Entrée							
Sortie							

Nœud	1	2	3	4	5	6	7
A							
B							
C							
D							
E							
F							
G							
Entrée							
Sortie							

Nœud	1	2	3	4	5	6	7
A							
B							
C							
D							
E							
F							
G							
Entrée							
Sortie							