POLYTECHNIQUE MONTRÉAL

Corrigé 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**	**20121**
Professeur	***Local***	***Téléphone***
Ettore Merlo – responsable / Tarek Ould Bachir - chargé	**M-5028**	**7128 / 5193**

Jour	*Date*	*Durée*	*Heures*
Mercredi	**30 avril 2012**	**2h30**	**9h30-12h00**

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.

Bonne chance à tous!

Important

Cet examen contient **6** questions sur un total de **15** 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

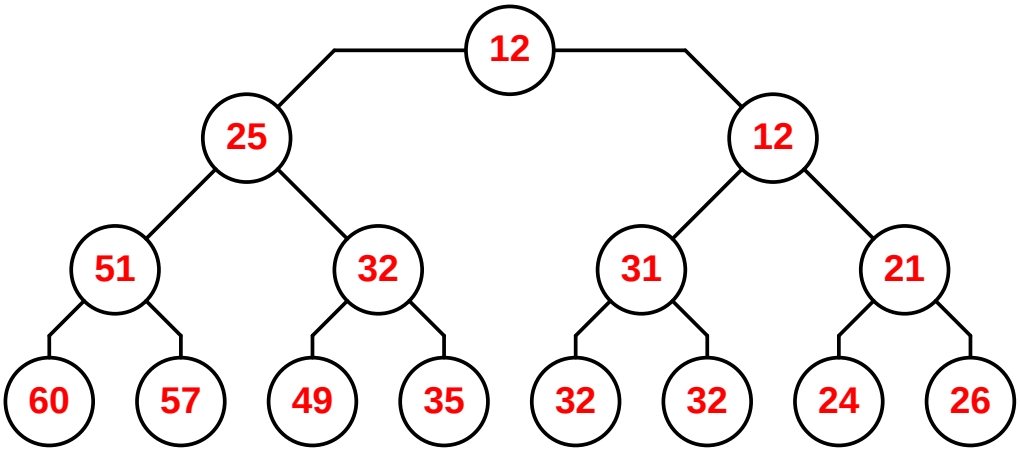
L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux (20 points)

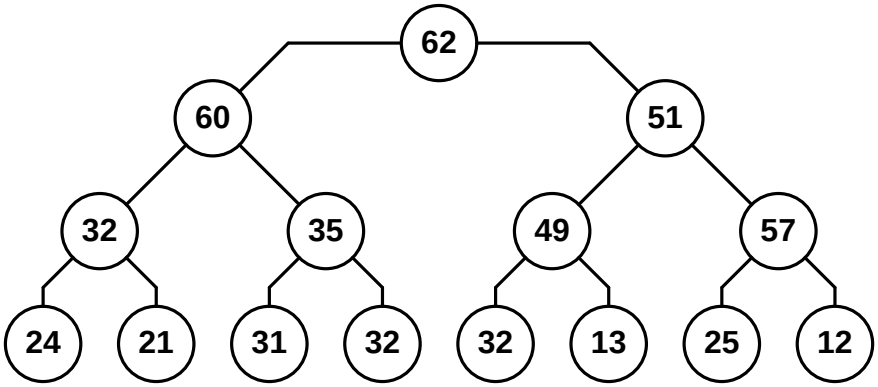
a) (5 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après :

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	-	12	25	13	51	32	31	21	60	57	49	35	32	32	24	26

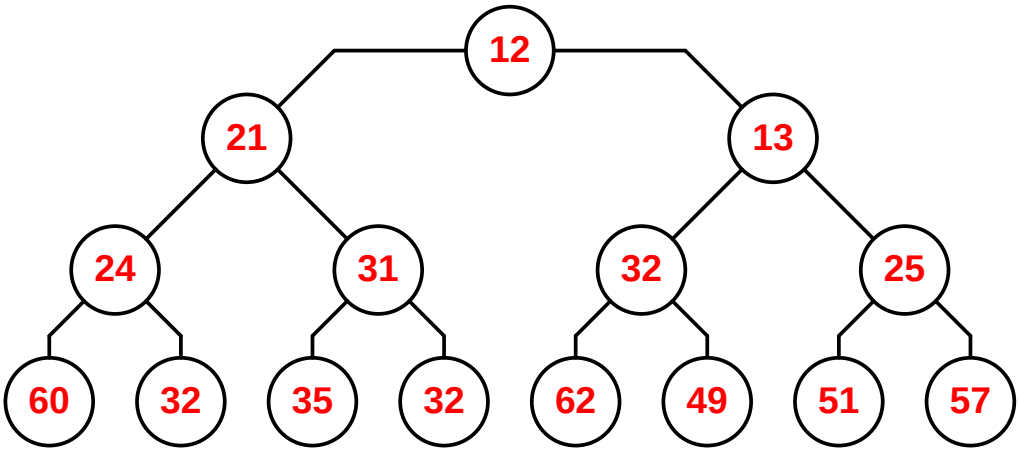
Votre réponse:



b) (7 pts) Construire, selon la technique vue en cours, un monceau MIN à partir de l'arbre binaire suivant :

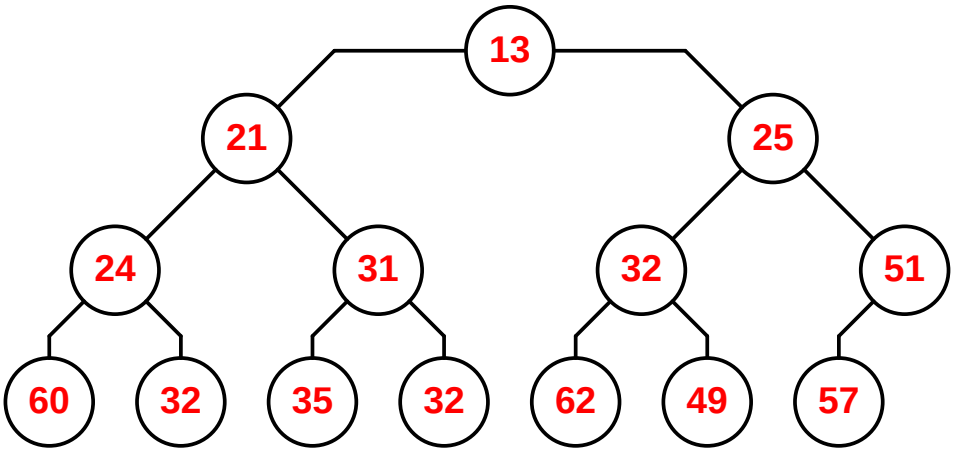


Monceau résultant :

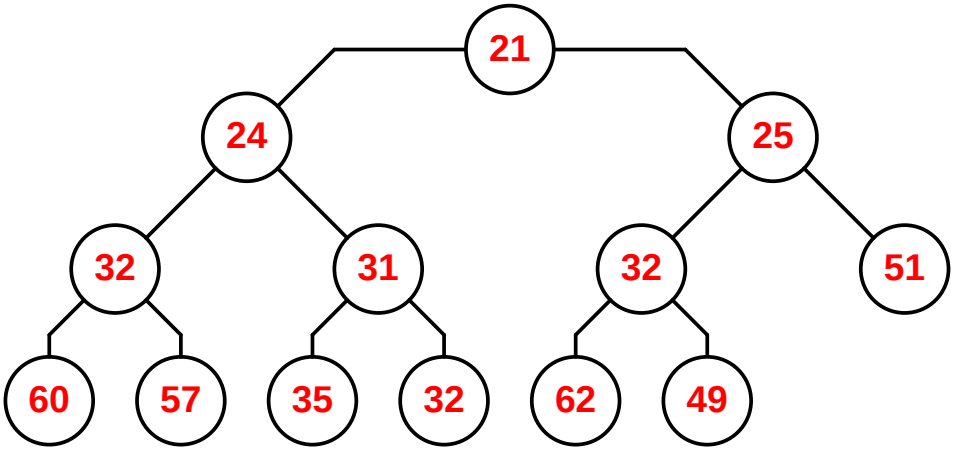


c) Dessiner l'état du monceau de la question 1.b) — monceau MIN que vous avez obtenu — suite à deux appels consécutifs à `deleteMin()` :

c.1) (1.5 pts) Monceau résultant du premier `deleteMin()`



c.2) (1.5 pts) Monceau résultant du second `deleteMin()`



d) (5 pts) Complétez la fonction `troisieme()`, qui retourne la troisième plus petite valeur d'un monceau MIN. Le code Java complet de l'implémentation du monceau MIN est donnée à l'Annexe 1 :

```
@SuppressWarnings("unchecked")
public AnyType troisieme() throws Exception
{
   if( currentSize < 3)
      throw new Exception("Le monceau a moins de 3 éléments");

   int maxIndex = Math.min(currentSize, 7); // COMPLÉTER ICI

   AnyType[] arr = (AnyType[]) new Comparable[3];

   arr[ 0 ] = array[ 1 ];

   arr[ 1 ] = (array[ 2 ].compareTo( array[ 3 ] ) < 0 ) ?
               array[ 2 ] : array[ 3 ];

   arr[ 2 ] = (array[ 2 ].compareTo( array[ 3 ] ) >= 0 ) ?
               array[ 2 ] : array[ 3 ];

   for(int i=4; i <= maxIndex; i++)
   {
      if( array[ i ].compareTo( arr[ 1 ] ) < 0 )
      {
          % COMPLÉTER ICI
          arr[ 2 ] = arr[ 1 ];
          arr[ 1 ] = array[ i ];



      }
      else if( array[ i ].compareTo( arr[ 2 ] ) < 0 )
      {
         % COMPLÉTER ICI
          arr[ 2 ] = array[ i ];




      }
   }

   return arr[2];
}
```

Question 2 : Recherche de patron (20 points)

On désire retrouver le patron P = "GAUGAUCHE" au moyen d'un automate.

a) (10 pts) Donnez la fonction de transition de l'automate à utiliser en complétant le tableau suivant. N'utiliser que les cases nécessaires.

État	A	C	E	G	H	U	Autre
0	0	0	0	1	0	0	0
1	2	0	0	1	0	0	0
2	0	0	0	1	0	3	0
3	0	0	0	4	0	0	0
4	5	0	0	1	0	0	0
5	0	0	0	1	0	6	0
6	0	7	0	4	0	0	0
7	0	0	0	1	8	0	0
8	0	0	9	1	0	0	0
9	0	0	0	1	0	0	0

b) (10 pts) Complétez le tableau suivant où l'on considère le texte T[1:26], et où l'on cherche à trouver l'état dans lequel se trouve votre automate après avoir reçu chacun des charactères de T.

Décalage s	Texte T[1:26]	État après la lecture	Décalages retenus
0	'G'	1	
1	'A'	2	
2	'C'	0	
3	'H'	0	
4	'I'	0	
5	'S'	0	
6	'A '	0	
7	'G'	1	

(continué)

(suite)

Décalage s	Texte T[1:26]	État après la lecture	Décalages retenus
8	'A'	**2**	
9	'U'	**3**	
10	'G'	**4**	
11	'A'	**5**	
12	'U'	**6**	
13	'G'	**4**	
14	'A'	**5**	
15	'U	**6**	
16	'G'	**4**	
17	'A'	**5**	
18	'U'	**6**	
19	'C'	**7**	
20	'H'	**8**	
21	'E'	**9**	√
22	'G'	**1**	
23	'A'	**2**	
24	'G'	**1**	
25	'E'	**0**	

Question 3 : Plus Longue Sous-séquence Commune (PLSC) (20 points)

a) (10 pts) En vous aidant du tableau suivant, donnez la PLSC des deux mots "PETOCHE" et "EPTOGRAPHE"

		P	**E**	**T**	**O**	**C**	**H**	**E**
	0	0	0	0	0	0	0	0
E	0	0, H	**1, D**	1, G	1, G	1, G	1, G	**1, D**
P	0	**1, D**	1, H	1, H	1, H	1, H	1, H	1, H
T	0	1, H	1, H	**2, D**	2, G	2, G	2, G	2, G
O	0	1, H	1, H	2, H	**3, D**	3, G	3, G	3, G
G	0	1, H	1, H	2, H	3, H	3, H	3, H	3, H
R	0	1, H	1, H	2, H	3, H	3, H	3, H	3, H
A	0	1, H	1, H	2, H	3, H	3, H	3, H	3, H
P	0	1, H	1, H	2, H	3, H	3, H	3, H	3, H
H	0	1, H	1, H	2, H	3, H	3, H	**4, D**	4, G
E	0	1, H	**2, D**	2, H	3, H	3, H	4, H	**5, D**

PLSC: **ETOHE**

Longueur de la PLSC: **5**

b) (5 pts) Si possible, proposez d'autres PLSC au couple de mots "PETOCHE" et "EPTOGRAPHE":

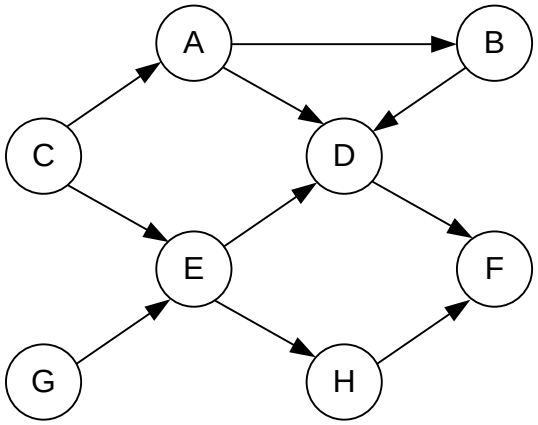
PTOHE

c) (5 pts) En utilisant les résultats tableau que vous avez obtenu en a), donnez l'ensemble des PLSC connues des mots "PETO" et "EPTO".

ETO, PTO

Question 4 : Ordre topologique (10 points)

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



Noeud	1	2	3	4	5	6	7	8
A	**1**	**0**						
B	**1**	**1**	**1**	**0**				
C	**0**							
D	**3**	**3**	**3**	**2**	**1**	**0**		
E	**2**	**1**	**0**					
F	**2**	**2**	**2**	**2**	**2**	**2**	**1**	**0**
G	**0**							
H	**1**	**1**	**1**	**1**	**0**			
Entrée	**C, G**	**A**	**E**	**B**	**H**	**D**	**-**	**F**
Sortie	**C**	**G**	**A**	**E**	**B**	**H**	**D**	**F**

Ordre trouvé:

Nœud	A	B	C	D	E	F	G	H
Ordre:	**3**	**5**	**1**	**7**	**4**	**8**	**2**	**6**

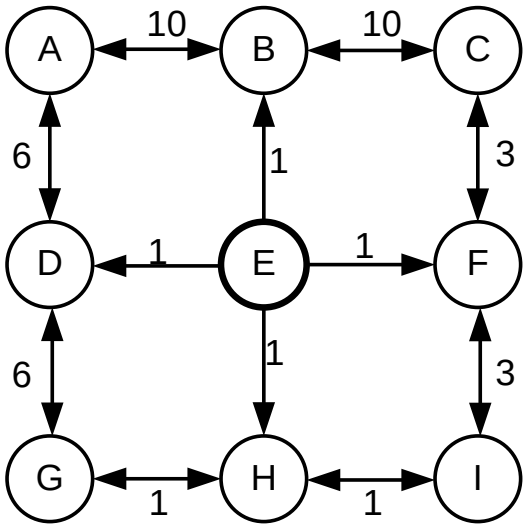
b) (3 pts) Proposez un ordre topologique alternatif à ce graphe:

Ordre alternatif:

Nœud	A	B	C	D	E	F	G	H
Ordre:	**3**	**5**	**2**	**7**	**4**	**8**	**1**	**6**

Question 5 : Algorithm de Dijkstra (14 points)

Considérez le graphe suivant. Les arcs à deux terminaisons indiquent des arcs double (de A à B et de B à A par exemple) de même poids.



a) (9 pts) Exécutez l'algorithme de Dijkstra utilisant une file de priorité pour trouver la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de E. Visitez les voisins d'un nœud par ordre alphabétique.

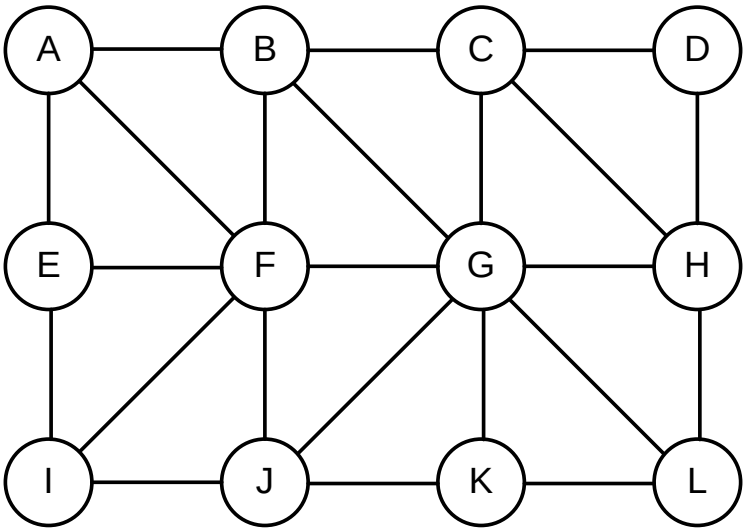
Nœud	Connu	Dist min.	Parent
A	√	∞, **11, 7**	**B, D**
B	√	∞, **1**	**E**
C	√	∞, **11, 4**	**B, F**
D	√	∞, **1**	**E**
E	√	**0,**	**-**
F	√	∞,**1**	**E**
G	√	∞, **7, 2**	**D, H**
H	√	∞, **1**	**E**
I	√	∞, **4, 2**	**F, H**

b) (5 pts) Détaillez chacun des chemins les plus courts trouvés parmi ceux demandés :

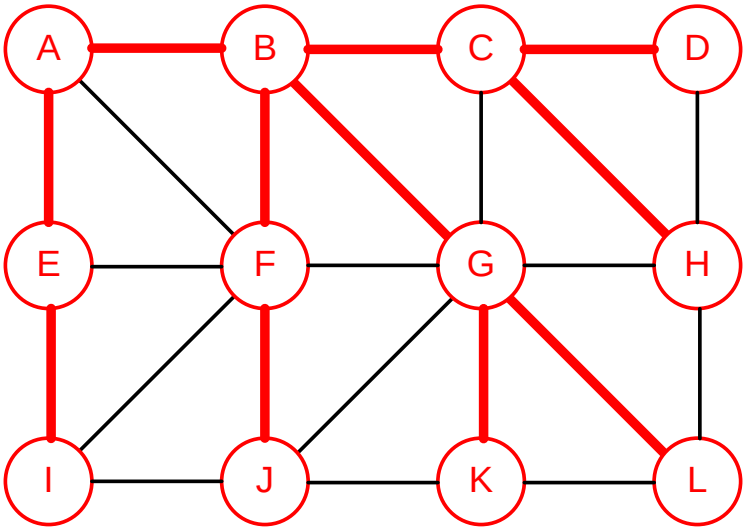
Destination	Le plus court chemin	Distance parcourue
A	**E → D → A**	**7**
C	**E → F → C**	**4**
G	**E → H → G**	**2**
I	**E → H → I**	**2**

Question 6 : Arbre sous-tendant minimum (16 points)

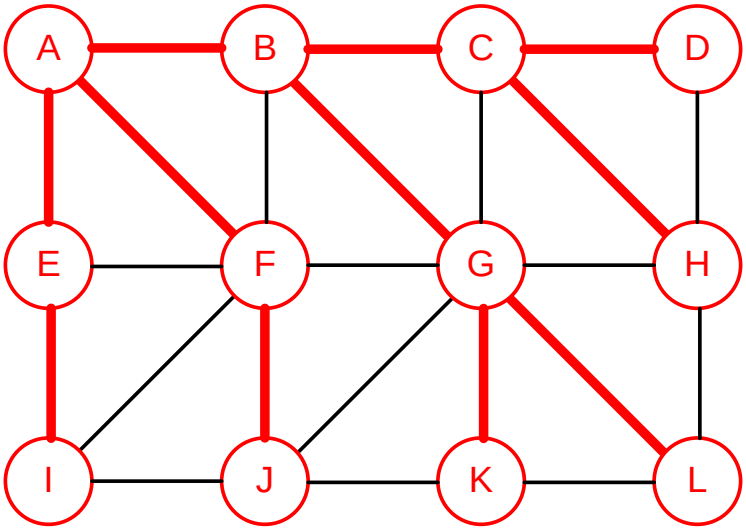
Considérez le graphe suivant dans lequel tous les arcs ont un poids unitaire.



a) (8 pts) Dessinez l'arbre sous-tendant minimum du graphe en utilisant l'algorithme de Prim. Partez du nœud A pour ce faire et visitez les voisins par ordre alphabétique.



b) (8 pts) Dessinez l'arbre sous-tendant minimum du graphe précédent en utilisant l'algorithme de Kruskal. Traitez les arcs par ordre alphabétique. Par exemple, vu du nœud E, on traitera (A, E) avant (E, F), et (E, F) avant (E, I).



Annexe 1

```
public class BinaryHeap<AnyType extends Comparable<? super AnyType>>
{
   public BinaryHeap( )
   {
      this( DEFAULT_CAPACITY );
   }

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

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

      int i = 1;
      for( AnyType item : items )
         array[ i++ ] = 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.compareTo( array[ hole / 2 ] ) < 0; hole /= 2 )
      array[ hole ] = array[ hole / 2 ];
      array[ hole ] = x;
   }

   @SuppressWarnings("unchecked")
   private void enlargeArray( int newSize )
   {
      AnyType [] old = array;
      array = (AnyType []) new Comparable[ newSize ];
      for( int i = 0; i < old.length; i++ )
      array[ i ] = old[ i ];
   }

   public AnyType findMin( )
   {
      if( isEmpty( ) )
      return null;
      return array[ 1 ];
   }
```

```
public AnyType deleteMin( )
{
   if( isEmpty( ) )
      return null;

   AnyType minItem = findMin( );
   array[ 1 ] = array[ currentSize-- ];
   percolateDown( 1 );

   return minItem;
}

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

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

public void makeEmpty( )
{
   currentSize = 0;
}

private static final int DEFAULT_CAPACITY = 10;

private int currentSize;      // Number of elements in heap
private AnyType [ ] array; // The heap array

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

   for( ; hole * 2 <= currentSize; hole = child )
   {
      child = hole * 2;
      if( child != currentSize &&
            array[ child + 1 ].compareTo( array[ child ] ) < 0 )
         child++;
      if( array[ child ].compareTo( tmp ) < 0 )
         array[ hole ] = array[ child ];
      else
         break;
   }
   array[ hole ] = tmp;
}
```

```
   @SuppressWarnings("unchecked")
   public AnyType troisieme() throws Exception
   {
      if( currentSize < 3)
         throw new Exception("Le monceau possède moins de 3 éléments");

      int maxIndex = // masqué pour la question;

      AnyType[] arr = (AnyType[]) new Comparable[3];

      arr[ 0 ] = array[ 1 ];

      arr[ 1 ] = (array[ 2 ].compareTo( array[ 3 ] ) < 0 ) ?
      array[ 2 ] : array[ 3 ];

      arr[ 2 ] = (array[ 2 ].compareTo( array[ 3 ] ) >= 0 ) ?
      array[ 2 ] : array[ 3 ];

      for( int i=4; i <= maxIndex; i++)
      {
         if( array[ i ].compareTo( arr[ 1 ] ) < 0 )
         {
            // masqué pour la question
         }
         else if( array[ i ].compareTo( arr[ 2 ] ) < 0 )
         {
            // masqué pour la question
         }
      }

      return arr[2];
   }
}
```