

Questionnaire examen final


INF2010

Sigle du cours

<i>Identification de l'étudiant(e)</i>		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20091
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Ettore Merlo – responsable / Tarek Ould Bachir - chargé		A-201	
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Samedi	25 avril 2009	2h30	9h30-12h00

<i>Documentation</i>	<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières	<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.

<i>Directives particulières</i>
<p> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Nous ne récupérerons pas le cahier supplémentaire à la fin de l'examen.</p> <p style="text-align: right;"><i>Bonne chance à tous!</i></p>

<i>Important</i>
<p>Cet examen contient 5 questions sur un total de 18 pages (excluant cette page)</p> <p>La pondération de cet examen est de 40 %</p> <p>Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux</p> <p>Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non</p>

Question 1 : Tri par monceau**(20 points)**

a) On désire étudier le comportement du tri par monceau.

a.1) (3 pts) Donnez la complexité algorithmique du tri par monceau (fonction du nombre d'éléments n) pour chacun des cas énumérés ci-dessous :

i) pire cas :

ii) cas moyen :

iii) meilleur cas :

a.2) (3 pts) Comparez les complexités du tri par monceau données dans **a.1)** à celles du tri par insertion et celles de QuickSort pour chacun des cas énumérés ci-dessous :

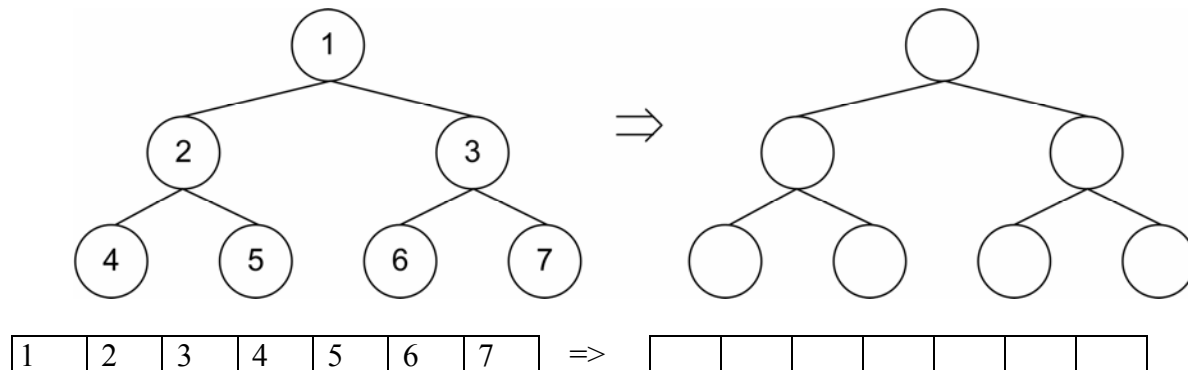
i) pire cas :

ii) cas moyen :

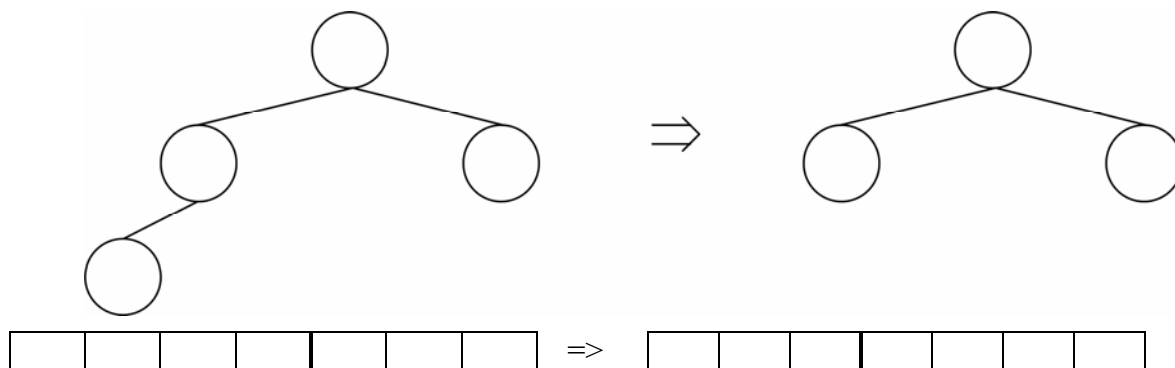
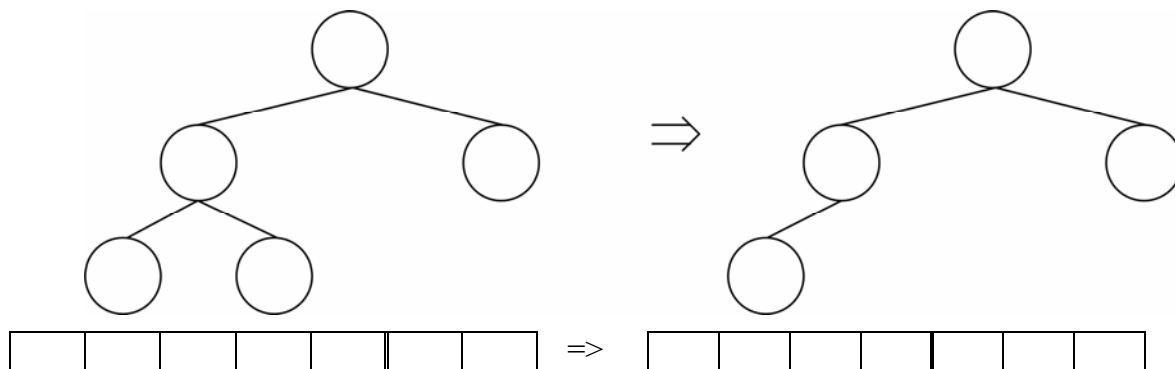
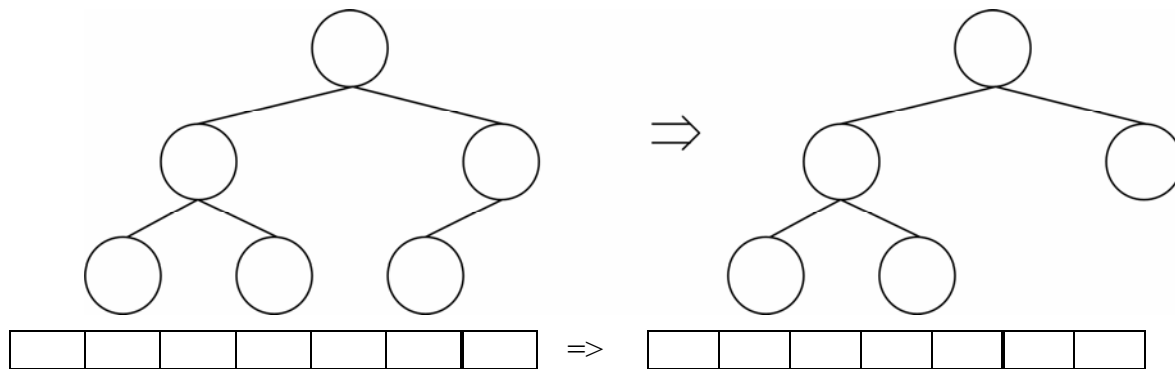
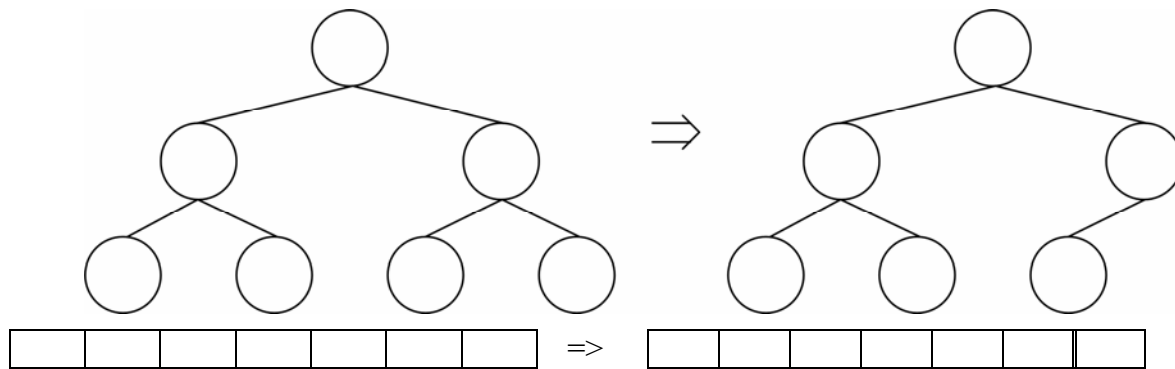
iii) meilleur cas :

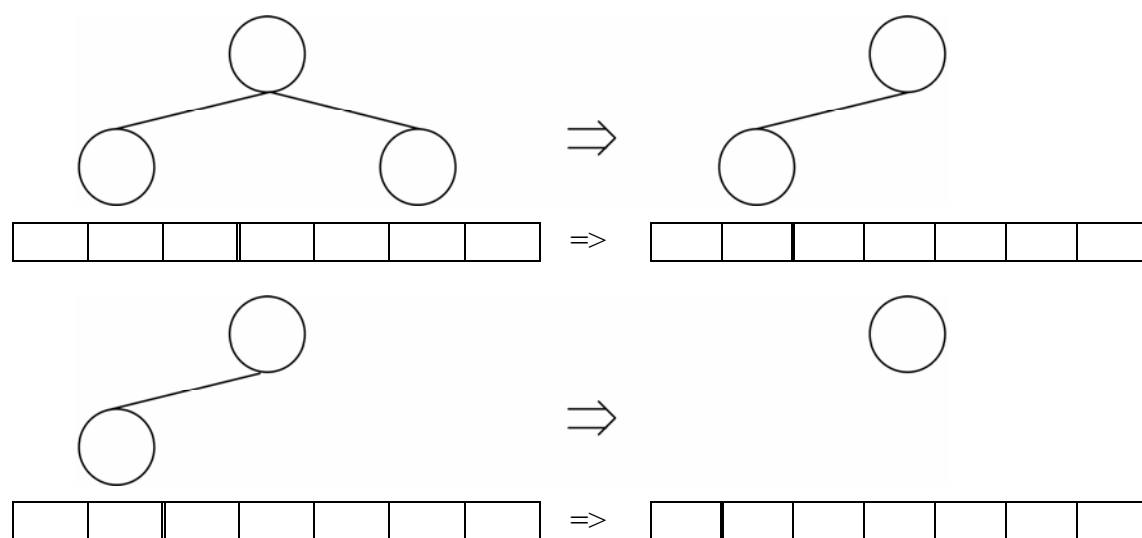
a.3) En partant du vecteur préalablement trié suivant $V = \{1, 2, 3, 4, 5, 6, 7\}$, on vous demande de :

i) **(5 pts)** construire un monceau max (indiquez l'état du vecteur après l'opération).



- ii) (6 pts) En partant du monceau max obtenu au point i), illustrez ci-dessous chacune des étapes du tri par monceau (indiquez l'état du vecteur avant et après l'opération).

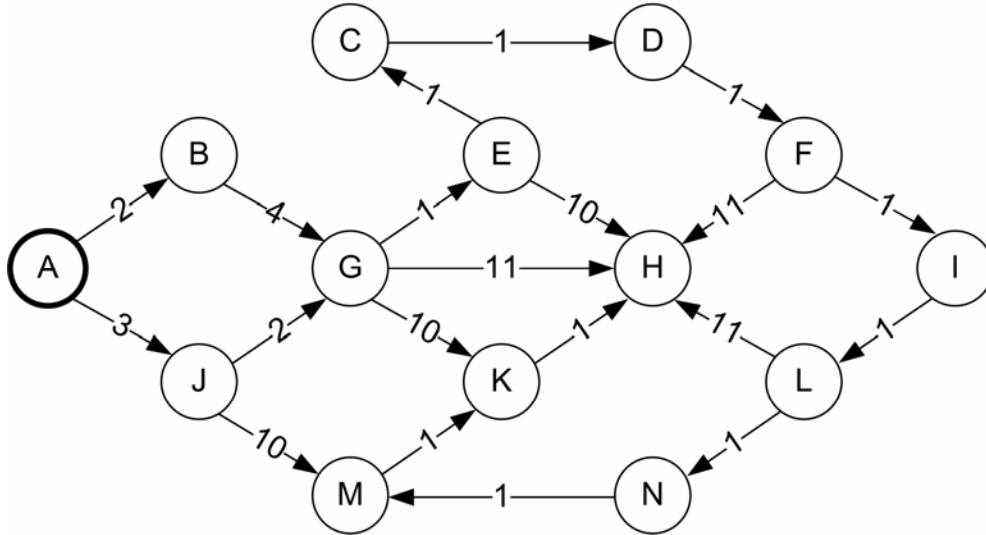




iii) (3 pts) Commentez le comportement de l'algorithme. Y a-t-il place à amélioration du tri par monceau selon vous ?

Question 2 : Plus court chemin d'un graphe acyclique**(20 points)****a) Tri topologique**

a.1) (7 pts) Pour le graphe dirigé suivant, numérotez les nœuds de **A** à **N** de sorte que, s'il existe un chemin du nœud **i** au nœud **j**, alors **i < j**.



Noeud	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A														
B														
C														
D														
E														
F														
G														
H														
I														
J														
K														
L														
M														
N														
Entrée														
Sortie														

a.2) (3 pts) Peut-on dire de ce graphe qu'il est acyclique ? Pourquoi ?

b) Nous voulons trouver les plus courts chemins depuis le nœud A jusqu'à l'ensemble des nœuds B à N en appliquant l'algorithme de Dijkstra.

b.1) (6 pts) Continuez l'exécution de l'algorithme de Dijkstra utilisant une file de priorité (ici elle est partiellement réalisée), pour trouver la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de A.

Nœud	Connu	Dist min.	Parent	File de priorité
A	√	0		D, 8
B	√	∞ , 2	A	M, 13
C	√	∞ , 7	E	K, 15
D		∞ , 8	C	H, 16
E	√	∞ , 6	G	
F		∞		
G	√	∞ , 6, 5	B, J	
H		∞ , 16	G	
I		∞		
J	√	∞ , 3	A	
K		∞ , 15	G	
L		∞		
M		∞ , 13	J	
N		∞		

b.2) (2 pts) Détaillez chacun des chemins les plus courts trouvés :

Nœud	Le plus court chemin	Distance parcourue
B	A → _____ → B	
C	A → _____ → C	
D	A → _____ → D	
E	A → _____ → E	
F	A → _____ → F	
G	A → _____ → G	
H	A → _____ → H	
I	A → _____ → I	
J	A → _____ → J	
K	A → _____ → K	
L	A → _____ → L	
M	A → _____ → M	
N	A → _____ → N	

b.3) (2 pts) Comparez le chemin $A \rightarrow J \rightarrow G \rightarrow E \rightarrow C \rightarrow D \rightarrow F \rightarrow I \rightarrow L \rightarrow N \rightarrow M$ avec le chemin allant de A à M que vous avez trouvé pour discuter du comportement de l'algorithme de Dijkstra.

Question 3 : Rabin-Karp**(20 points)**

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 nombre qu'il faut pré-calculer et on compare toutes les valeurs des sous-séquences du texte à la valeur pré-calculée. Les différentes implémentations (**code java**) discutées dans cette question vous sont données à l'**annexe I**.

Dans ces implémentations de Rabin-Karp, le texte et le patron ne contiennent que les caractères alphabétiques 'a'-'z', 'A'-'Z' et l'espace ' ' dans l'encodage ASCII. Les valeurs respectives de ces caractères sont données ci-dessous :

Caractère	Val. ASCII	Caractère	Val. ASCII
'A'	65	'a'	97
'B'	66	'b'	98
...		...	
'Y'	89	'y'	121
'Z'	90	'z'	122
Espace	32	-	-

On vous propose la méthode de calcul suivante :

Polynôme modulaire dans la base $d = 10$ modulo 11

Par exemple, $P = \text{'Anna'}$: Code : 65, 110, 110, 97

$$\begin{aligned}
 p &= (((((65\%11) * 10 + 110) \% 11) * 10 + 110) \% 11) * 10 + 97) \% 11 \\
 &= (((((10) * 10 + 110) \% 11) * 10 + 110) \% 11) * 10 + 97) \% 11 \\
 &= (((((210) \% 11) * 10 + 110) \% 11) * 10 + 97) \% 11 \\
 &= (((1 * 10 + 110) \% 11) * 10 + 97) \% 11 \\
 &= (((120) \% 11) * 10 + 97) \% 11 \\
 &= ((10) * 10 + 97) \% 11 \\
 &= (197) \% 11 \\
 &= 10
 \end{aligned}$$

a) (4 pts) Donnez la valeur de p pour le patron

$P = \text{'merlo'}$: Codes ASCII : 109, 101, 114, 108, 111

b) (16 pts) Retrouvez tous les décalages donnant la présence de P dans le texte

T= « Un amerloque se divertit »

Codes ASCII :

U	n		a	m	e	r	l	o	q	u	e		s	e		d	i	v	e	r	t	i	t
85	110	32	97	109	101	114	108	111	113	117	101	32	115	101	32	100	105	118	101	114	116	105	116

Décalage (s)	Sous-chaine	Valeur du polynôme	Égalité?	Faux positif?	Correspondance?
0	‘Un am’				
1	‘n ame’				
2	‘ amer’				
3	‘amerl’				
4	‘merlo’				
5	‘erloq’				
6	‘rloqu’				
7	‘loque’				
8	‘oque ’				
9	‘que s’				
10	‘ue se’				
11	‘e se ’				
12	‘ se d’				
13	‘se di’				
14	‘e div’				
15	‘ dive’				
16	‘diver’				
17	‘ivert’				
18	‘verti’				
19	‘ertit’				

Faux positifs trouvés :

Décalages retournés :

Question 4 : PLSC**(20 points)**

Partant de la table de *PLSC* pour les entrées ACRYLONITRILE et ANTICLERICAL suivante, donnant ANTILE pour solution au problème, on propose :

	A	C	R	Y	L	O	N	I	T	R	I	L	E
	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1	1	1	1	1
N	0	1	1	1	1	1	2	2	2	2	2	2	2
T	0	1	1	1	1	1	2	2	3	3	3	3	3
I	0	1	1	1	1	1	2	3	3	3	4	4	4
C	0	1	2	2	2	2	2	3	3	3	4	4	4
L	0	1	2	2	3	3	3	3	3	3	4	5	5
E	0	1	2	2	3	3	3	3	3	3	4	5	6
R	0	1	2	3	3	3	3	3	3	4	4	5	6
I	0	1	2	3	3	3	3	4	4	4	5	5	6
C	0	1	2	3	3	3	3	4	4	4	5	5	6
A	0	1	2	3	3	3	3	4	4	4	5	5	6
L	0	1	2	3	4	4	4	4	4	4	5	6	6

- a) d'évaluez la *PLSC* pour les entrées NITRILE et ANTICLERICAL en tronquant la première table et de récupérer le résultat partiel NTILE :

	N	I	T	R	I	L	E
	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1
N	0	2	2	2	2	2	2
T	0	2	3	3	3	3	3
I	0	2	3	3	4	4	4
C	0	2	3	3	4	4	4
L	0	3	3	3	4	5	5
E	0	3	3	3	4	5	6
R	0	3	3	4	4	5	6
I	0	3	4	4	5	5	6
C	0	3	4	4	5	5	6
A	0	3	4	4	5	5	6
L	0	4	4	4	5	6	6

a.1) (2 pts) Est-ce que NTILE est une *PLSC* des entrées NITRILE et ANTICLERICAL ?

a.2) (2 pts) Discutez brièvement la validité du procédé de troncature dans ce cas de figure.

a.3) (6 pts) Proposez une autre *PLSC* des entrées NITRILE et ANTICLERICAL si elle existe (vous pouvez vous servir de la table donnée à l'annexe II).

b) d'évaluez la *PLSC* pour les entrées ACRYLONITRILE et CLERICAL en tronquant la première table et de récupérer le résultat partiel LE :

	A	C	R	Y	L	O	N	I	T	R	I	L	E
C	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	↑ 1	↖ 2	← 2	← 2	← 2	↑ 2	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4
E	0	↑ 1	↑ 2	↑ 2	↑ 2	↖ 3	← 3	← 3	↑ 3	↑ 3	↑ 3	↑ 4	↖ 5
R	0	↑ 1	↑ 2	↑ 2	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 5
I	0	↑ 1	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 5	↑ 6
C	0	↑ 1	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6
A	0	↑ 1	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6
L	0	↑ 1	↑ 2	↑ 3	↑ 3	↖ 4	← 4	← 4	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6

b.1) (2 pts) Est-ce que LE est une *PLSC* des entrées ACRYLONITRILE et CLERICAL ?

b.2) (2 pts) Discutez brièvement la validité du procédé de troncature dans ce cas de figure.

b.3) (6 pts) Proposez une autre *PLSC* des entrées ACRYLONITRILE et CLERICAL si elle existe (vous pouvez vous servir de la table donnée à l'annexe III) :

Question 5 : Ensemble disjoints**(20 points)**

Partant du théorème suivant :

Tout ensemble S_n , défini comme l'ensemble des n premiers carrés entiers, admet une partition parfaite de quatre (4) ensembles disjoints (notés $S_{n,A}$, $S_{n,B}$, $S_{n,C}$, $S_{n,D}$) si et seulement si $n \geq 15$ et que $n \bmod 8 = 0$ ou $n \bmod 8 = 7$; cette partition est telle que chacune des sommes des éléments de ces ensembles disjoints sont égales.

Exemple :

Pour $n=15$, $S_{15}=\{1^2, 2^2, 3^2, 4^2, 5^2, 6^2, 7^2, 8^2, 9^2, 10^2, 11^2, 12^2, 13^2, 14^2, 15^2\}$; nous avons une partition parfaite (elle est aussi unique) :

$$S_{15,A} = \{1^2, 7^2, 8^2, 14^2\}$$

$$S_{15,B} = \{2^2, 9^2, 15^2\}$$

$$S_{15,C} = \{3^2, 6^2, 11^2, 12^2\}$$

$$S_{15,D} = \{4^2, 5^2, 10^2, 13^2\}$$

Où la somme des éléments de chaque ensemble disjoint est 310.

- a) (2 pts) Peut-on trouver une partition de 4 ensembles disjoints à somme égale pour les ensembles S_n suivants :

S_n	Partition existe	Partition n'existe pas
S_{15}	✓	
S_{16}	✓	
S_{23}		
S_{27}		
S_{33}		
S_{48}		

Justifiez vos réponses :

b) Sachant que :

L'ensemble des 32 carrés consécutifs $C_i = S_{i+31} - S_{i-1} = \{i^2, (i+1)^2, \dots, (i+31)^2\}$ ($i \geq 1$) admet une partition parfaite unique :

$$C_{i,A} = \{i^2, (i+7)^2, (i+11)^2, (i+12)^2, (i+18)^2, (i+21)^2, (i+25)^2, (i+30)^2\}$$

$$C_{i,B} = \{(i+1)^2, (i+6)^2, (i+10)^2, (i+13)^2, (i+19)^2, (i+20)^2, (i+24)^2, (i+31)^2\}$$

$$C_{i,C} = \{(i+2)^2, (i+5)^2, (i+9)^2, (i+14)^2, (i+16)^2, (i+23)^2, (i+27)^2, (i+28)^2\}$$

$$C_{i,D} = \{(i+3)^2, (i+4)^2, (i+8)^2, (i+15)^2, (i+17)^2, (i+22)^2, (i+26)^2, (i+29)^2\}$$

Où la somme des éléments de chaque ensemble disjoint est $8i^2 + 248i + 2604$.

b.1) (2 pts) Proposez une partition parfaite pour S_{32} :

	Éléments							
$S_{32,A}$								
$S_{32,B}$								
$S_{32,C}$								
$S_{32,D}$								

b.2) (3 pts) Combien existe-t-il de partitions parfaites pour S_{47} ? Justifiez par un calcul.

$$\text{Indice : } 47 = 15 + 32$$

b.3) (3 pts) En vous servant du raisonnement utilisé pour **b.2)**, proposez une partition parfaite pour S_{47} :

	Éléments											
$S_{47,A}$												
$S_{47,B}$												
$S_{47,C}$												
$S_{47,D}$												

c) On vous demande d'exécuter la structure d'ensembles disjoints utilisant la compression du chemin *et* la métrie des rangs¹ pour générer la partition de S_{32} (**b.I**).

c.I) (4 pts) Dessinez la forêt d'arbres (indiquez le rang de chaque arbre) obtenue à la fin des appels suivants (pseudo-code java) :

```
// A la fin de cet appel, nous avons 32 arbres
for(int i=1; i ≤ 32; i++)
    Create( $i^2$ );

// A la fin de ces appels, nous avons 16 arbres
Union(  $1^2$ ,  $8^2$ ); Union( $12^2$ ,  $13^2$ ); Union( $19^2$ ,  $22^2$ ); Union( $26^2$ ,  $31^2$ );
Union(  $2^2$ ,  $7^2$ ); Union( $11^2$ ,  $14^2$ ); Union( $20^2$ ,  $21^2$ ); Union( $25^2$ ,  $32^2$ );
Union(  $3^2$ ,  $6^2$ ); Union( $10^2$ ,  $15^2$ ); Union( $17^2$ ,  $24^2$ ); Union( $28^2$ ,  $29^2$ );
Union(  $4^2$ ,  $5^2$ ); Union(  $9^2$ ,  $16^2$ ); Union( $18^2$ ,  $23^2$ ); Union( $27^2$ ,  $30^2$ );

// A la fin de ces appels, nous avons 8 arbres
Union(  $1^2$ ,  $12^2$ ); Union( $19^2$ ,  $26^2$ ); Union( $11^2$ ,  $20^2$ ); Union( $20^2$ ,  $25^2$ );
Union(  $3^2$ ,  $17^2$ ); Union(  $3^2$ ,  $28^2$ ); Union(  $4^2$ ,  $9^2$ ); Union( $18^2$ ,  $27^2$ );
```

N.B.: Vous pouvez omettre le carré dans les nœuds et représenter le nœud i^2 par i

ARBRE 1	ARBRE 2	ARBRE 3	ARBRE 4
Rang :	Rang :	Rang :	Rang :
ARBRE 5	ARBRE 6	ARBRE 7	ARBRE 8
Rang :	Rang :	Rang :	Rang :

¹ **Rappel :** i) Lors de l'union de deux arbres de rangs différents, l'arbre de rang inférieur devient l'enfant de l'arbre de rang supérieur. Le nouveau rang est le plus grand des deux rangs.
 ii) La racine de l'arbre issu de l'union de deux arbres de même rang est la racine ayant la plus petite valeur parmi les deux arbres. Le nouveau rang est celui des deux arbres incrémenté de 1.

c.2) (4 pts) En partant des résultats de la question **c.1)**, dessinez la forêt d'arbres (indiquez le rang de chaque arbre) obtenue à la fin des appels :

// A la fin de ces appels, nous avons 4 arbres

`Union(12, 312); Union(72, 322); Union(152, 242); Union(42, 302);`

N.B.: Vous pouvez omettre le carré dans les nœuds et représenter le nœud i^2 par i

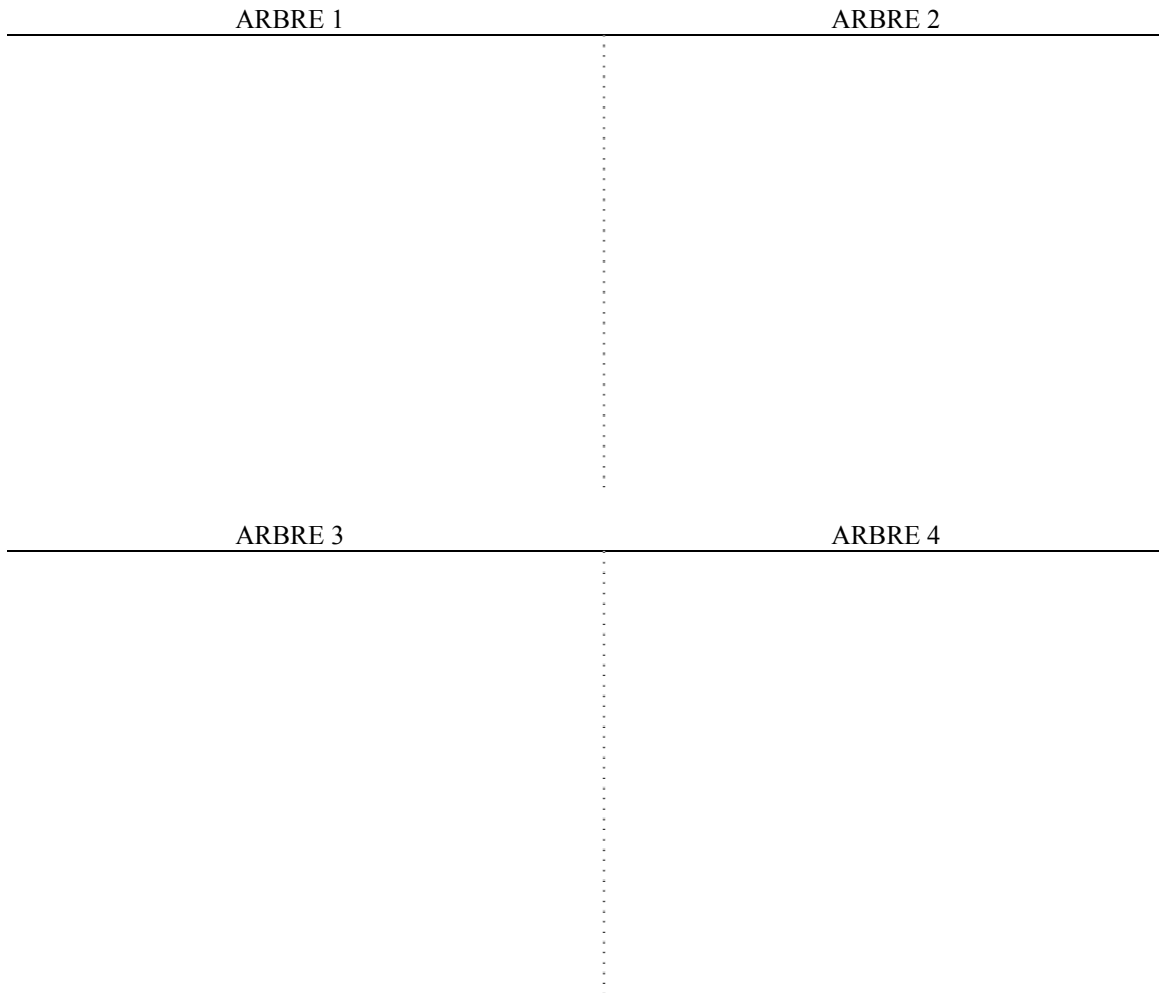
ARBRE 1	ARBRE 2
<div style="border: 1px solid black; padding: 5px; margin-top: 10px;">Rang :</div>	<div style="border: 1px solid black; padding: 5px; margin-top: 10px;">Rang :</div>
ARBRE 3	ARBRE 4
<div style="border: 1px solid black; padding: 5px; margin-top: 10px;">Rang :</div>	<div style="border: 1px solid black; padding: 5px; margin-top: 10px;">Rang :</div>

c.3) (2 pts) En partant des résultats de la question **c.2)**, dessiner la forêt d'arbres obtenue à la fin des appels :

// A la fin de ces appels, nous avons 4 arbres

Find(31^2); **Find**(21^2); **Find**(14^2); **Find**(29^2), **Find**(30^2);

N.B.: Vous pouvez omettre le carré dans les nœuds et représenter le nœud i^2 par i



Annexe I

```

import java.util.ArrayList;

public class RabinKarp {

    public static ArrayList<Integer>
    RabinKarpFind(String Text, String Pattern)
    {
        ArrayList<Integer> decalages = new ArrayList<Integer>();

        if( Text.length() < Pattern.length() )
            return decalages;

        int p = ComputePatternValue( Pattern );

        for(int i=0; i <= Text.length() - Pattern.length(); i++)
        {
            int t = ComputePatternValue(Text.substring(i, i+Pattern.length()));

            if( t == p )
            {
                int j;
                for(j=0; j< Pattern.length(); j++)
                {
                    if( Pattern.charAt( j ) != Text.charAt( i +j ) )
                        break;
                }

                if(j == Pattern.length())
                {
                    decalages.add( i );
                    System.out.println("Correspondance à " + i);
                }
                else
                    System.out.println("Faux positif à " + i);
            }
        }

        return decalages;
    }

    public static int ComputePatternValue(String Pattern)
    {
        int p = 0;

        for(int i=0; i<Pattern.length(); i++)
        {
            p *=10;
            p += (int) Pattern.charAt( i );
            p %= 11;
        }

        return p;
    }

    public static void main(String[] args)
    {
        String Pattern = "merlo";
        int p = ComputePatternValue(Pattern);
        System.out.println( p );

        String Text = "Un amerloque se divertit";
        System.out.println( Text.length() );
        ArrayList<Integer> decalages = RabinKarpFind(Text, Pattern);

        for(int s : decalages )
            System.out.print( s + "\t" );
    }
}

```

Annexe II

a) *PLSC* de NITRILE et ANTICLERICAL

		N	I	T	R	I	L	E
	0	0	0	0	0	0	0	0
A	0							
N	0							
T	0							
I	0							
C	0							
L	0							
E	0							
R	0							
I	0							
C	0							
A	0							
L	0							

Annexe III

b) *PLSC* de ACRYLONITRILE et CLERICAL :

		A	C	R	Y	L	O	N	I	T	R	I	L	E
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0													
L	0													
E	0													
R	0													
I	0													
C	0													
A	0													
L	0													