ÉCOLE POLYTECHNIQUE MONTRÉAL

Le génie sans frontières

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**	**20093**
Professeur	*Local*	*Téléphone*
T. Lavoie et T. Ould Bachir (chargés de cours) Pr. Ettore Merlo (coordonateur)	**B-415 (gr. 1) B-418 (gr. 2)**	

Jour	*Date*	*Durée*	*Heures*
Lundi	**07 décembre 2009**	**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. Nous ne récupérerons pas le cahier supplémentaire à la fin de l'examen.

Bonne chance à tous!

Important

Cet examen contient **8** questions sur un total de **13** pages **(incluant 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

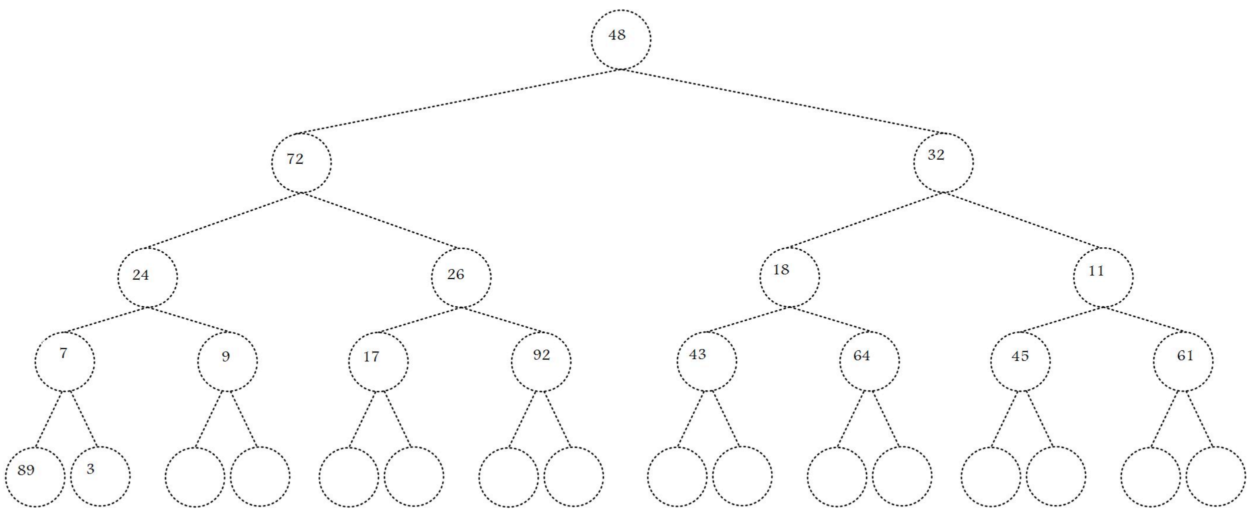
L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux (10 points)

1.1) (2 points) Considérez le tableau suivant qui représente un arbre binaire:

	48	72	32	24	26	18	11	7	9	17	92	43	64	45	61	89	3														

Indiquez l'emplacement de chacun des éléments dans l'arbre ci-dessous



1.2) (2 points) L'arbre de la question 1.1) est-il un monceau?

Non. On le voit par exemple au fait que 3 est l'enfant de 7.

1.3) (2 points) Donnez la formule pour calculer l'index des enfants droit et gauche du nœud à l'index i.

Enfant droit: **2i+1** Enfant gauche: **2i**

1.4) (2 points) Si N est le nombre de nœuds dans un monceau, combien de nœuds faut-il vérifier au maximum pour trouver le nœud maximum dans un monceau MIN ?

Le maximum est forcément une feuille. Il est impossible de savoir quelle feuille contient le maximum. Il faut donc vérifier toutes les feuilles. Il y a au plus (N+1)/2 (division entière) feuilles. La réponse est donc (N+1)/2 (division entière).

1.5) (2 points) Quelle est la complexité asymptotique de la construction d'un monceau?

La construction d'un monceau s'effectue en O(N).

Question 2 : Tri par monceau (15 points)

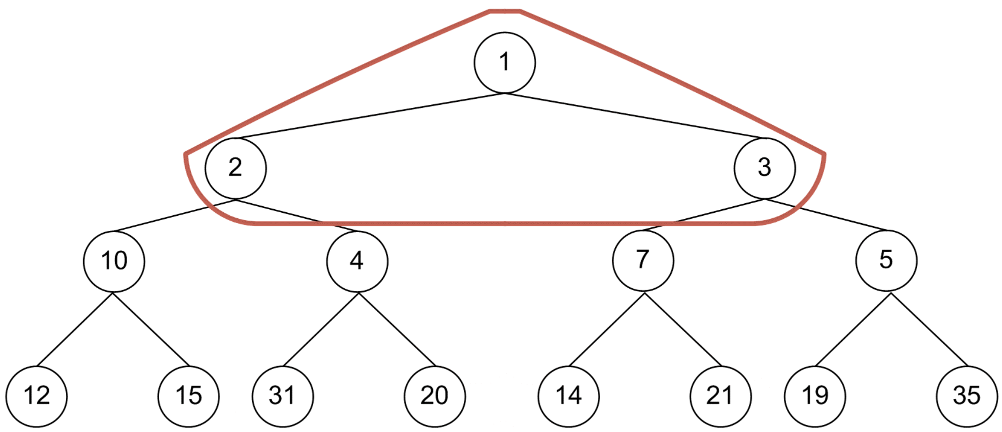
On désire trouver les trois plus petits éléments d'un vecteur d'entiers. On vous suggère, pour ce faire, d'utiliser un monceau MIN et de trouver les trois plus petits éléments au niveau de la racine et de ses deux enfants.

2.1) (3 points) Commentez cette stratégie. Est-elle justifiée ou non? Expliquez clairement votre réponse.

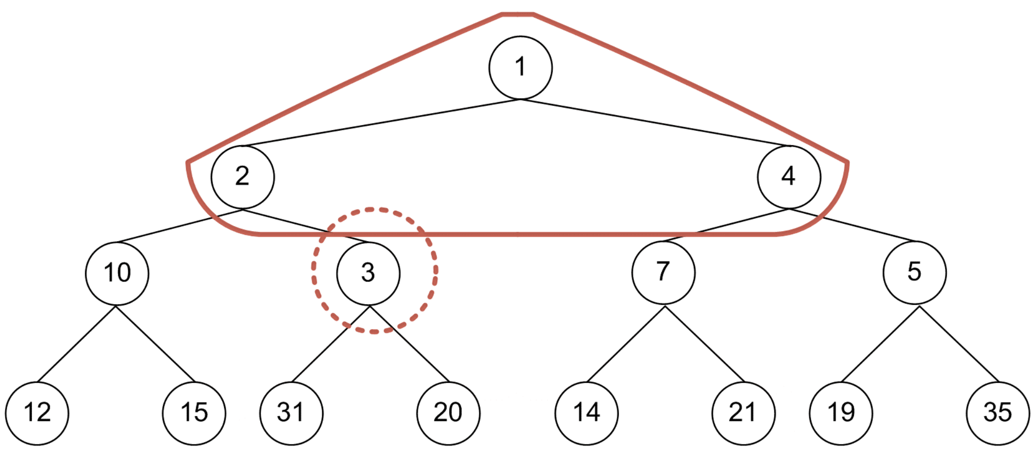
C'est une mauvaise stratégie car elle ne réussit pas à tous les coups.

2.2) Illustrez votre réponse en construisant un monceau min de chacun des vecteurs suivants :

2.2.1) (1.5 points) 15, 31, 21, 10, 20, 14, 5, 12, 2, 4, 1, 3, 7, 19, 35



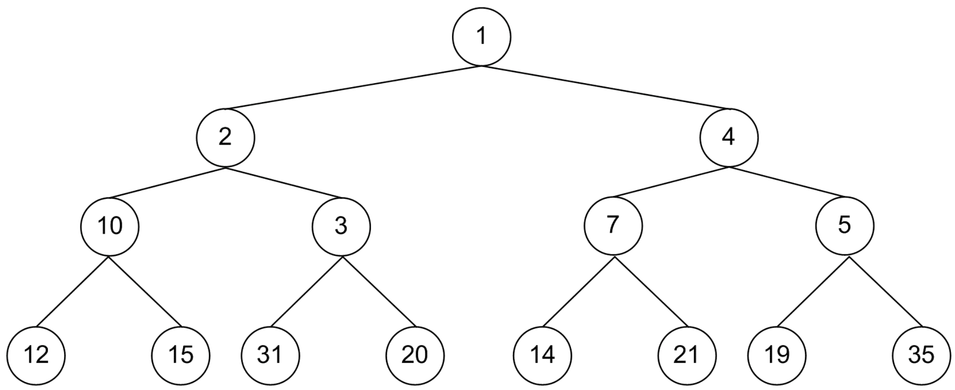
2.2.2) (1.5 points) 15, 31, 21, 10, 20, 14, 5, 12, 2, 3, 1, 4, 7, 19, 35



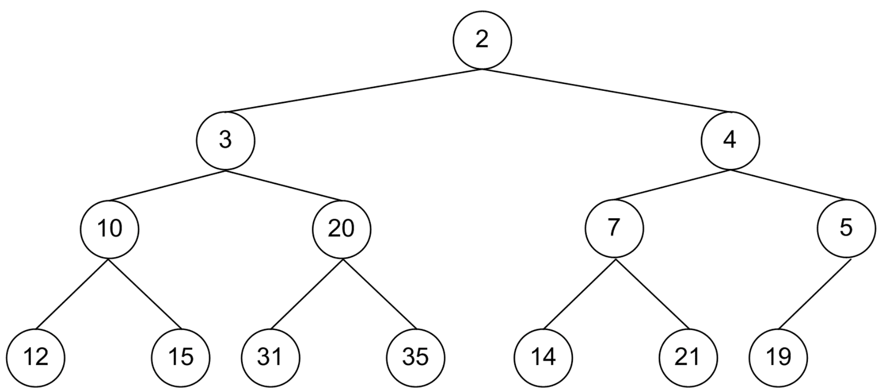
On voit donc bien que cette stratégie ne réussit pas toujours.

2.3) Exécutez l'algorithme de tri partiel vu en cours pour trouver les trois plus petites valeurs parmi les entiers 15, 31, 21, 10, 20, 14, 5, 12, 2, 3, 1, 4, 7, 19, 35 (ceux de 2.2.2). Pour mémoire, le principe du tri partiel consiste à construire un monceau MIN et à retire la racine pour chaque nouvelle valeur.

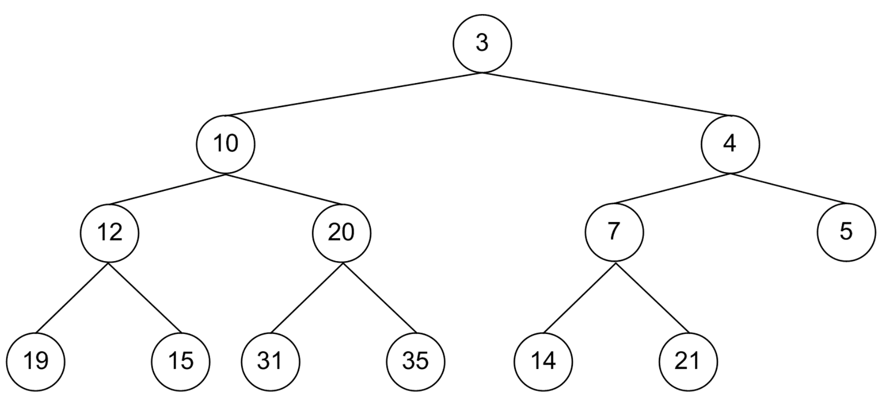
2.3.1) (3 points) Premier monceau du tri partiel, avant retrait de la racine :



2.3.2) (3 points) Second monceau du tri partiel, avant retrait de la racine :



2.3.3) (3 points) Troisième monceau du tri partiel, avant retrait de la racine :



Question 3 : Modélisation par graphes (15 points)

3.1) (5 points) Soit un labyrinthe composés d'un nombre N de salles. Chaque salle est de forme carrée et peut avoir jusqu'à 4 portes conduisant à une salle adjacente, soit au nord, au sud, à l'est ou à l'ouest. Proposez une façon de modéliser ce problème avec un graphe. Indiquez clairement ce que représente les nœuds et les arrêtes.

Chaque sommet du graphe représente une salle du labyrinthe. Les arrêtes représentent les portes et il y en a une entre chaques salles adjacentes non-séparées par un mur. Le graphe n'est pas orienté.

3.2) (5 points) Supposons que l'on vous donne un graphe G(V, E) qui respecte votre modélisation de la question 3.1). Donnez, en notation asymptotique, le temps d'exécution le plus rapide qu'il est possible d'obtenir pour le problème consistant à trouver la sortie du labyrinthe à partir du point de départ.

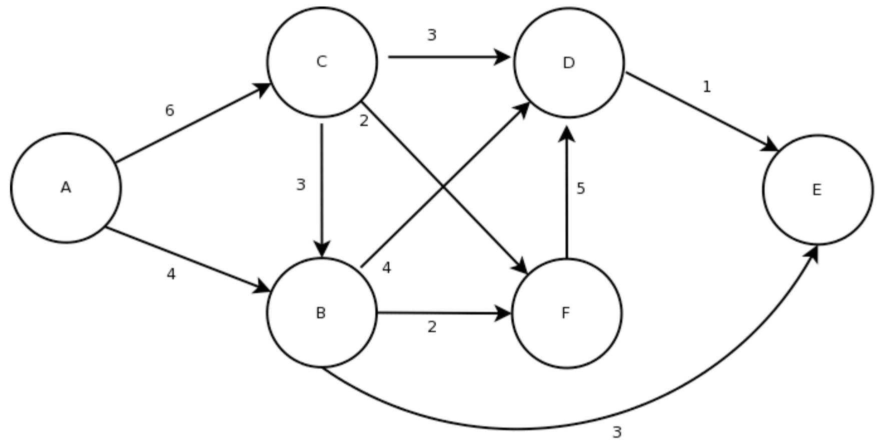
Ce problème est équivalent à trouver le plus court chemin entre deux sommets d'un graphe dans un graphe non-valué. Comme le graphe est éparse ($|E| < 4\times|V|$), l'algorithme pour résoudre ce problème a une complexité de $O(|E| + |V|)$.

3.3) (5 points) Supposons maintenant que l'on permette de détruire des murs pour trouver un chemin, mais la destruction entraîne une pénalité de $P > 0$ cases. Quelles modifications faut-il apporter à celle proposée en 3.1) ? Comment peut-on trouver le chemin le moins pénalisant dans cette version du problème du labyrinthe ?

Il suffit de transformer le graphe en a en un graphe valué où le coût entre deux nœuds est égal à la pénalité entre deux cases adjacente (0 s'il n'y a pas de murs). Le problème devient alors équivalent à trouver le plus court chemin dans un graphe valué; ce problème résoluble par l'algorithme de Dijkstra.

Question 4 : Algorithme de Dijkstra (15 points)

4.1) (10 points) Soit le graphe suivant:



Calculez à l'aide de l'algorithme de Dijkstra (n'utilisant pas de monceau) la distance minimale entre le point A et tous les autres points dans le graphe. Donnez les étapes de votre exécution de l'algorithme dans la table qui suit.

Nœud	Distance A	Distance B	Distance C	Distance D	Distance E	Distance F
-	0	inf	inf	inf	inf	inf
A	**0**	**4**	**6**	**inf**	**inf**	**inf**
B	**0**	**4**	**6**	**8**	**7**	**6**
C	**0**	**4**	**6**	**8**	**7**	**6**
F	**0**	**4**	**6**	**8**	**7**	**6**
E	**0**	**4**	**6**	**8**	**7**	**6**
D	**0**	**4**	**6**	**8**	**7**	**6**

4.2) (5 points) Quelle est la complexité en pire cas de l'algorithme de Dijkstra que vous venez d'exécuter ? Sachant que l'algorithme peut être amélioré, dîtes comment et donnez la nouvelle complexité ?

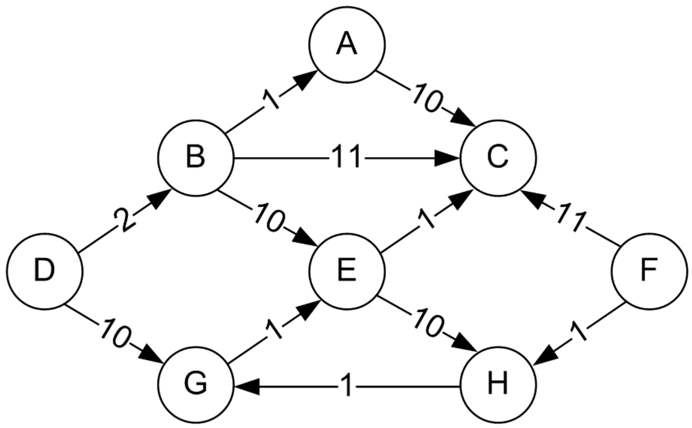
Pire cas: O(|V^2|)

L'algorithme peut être amélioré en utilisant un monceau. La complexité devient alors O(|E|log(V)).

Question 5 : Ordre topologique (10 points)

Pour chacun des graphes dirigés aux arêtes pondérées suivants, dîtes si le graphe admet un ordre topologique et si oui proposez en un. Vous pouvez vous aider des grilles données à l'annexe I. Identifiez clairement cette annexe à votre nom et matricule et assurez-vous de la remettre avec votre copie (même si elle demeure vierge).

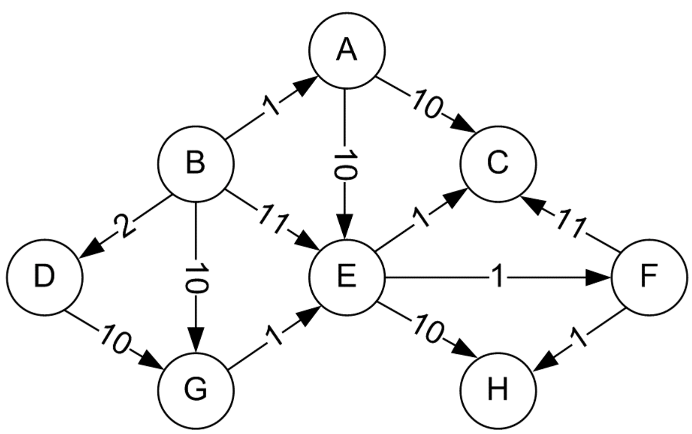
5.1) (5 points)



Ce graphe est cyclique est n'admet pas d'ordre topologique

A	B	C	D	E	F	G	H
×	×	×	×	×	×	×	×

5.2) (5 points)



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

Question 6 : Rabin-Karp (15 points)

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. Supposons que les chaînes de caractères soient constituées des chiffres 0 à 9 auxquels on associe la valeur numérale de ces derniers (le chiffre 9 vaut 9).

6.1) (2 points) Quel est le nombre de faux-positifs que produit l'exécution de l'algorithme Rabin-Karp sur la chaîne de caractères 96325874124896323495 si l'on recherche le patron 96 en travaillant modulo q=11 ?

Note : Vous n'êtes pas obligé de donner vos calculs intermédiaires. On vous conseille cependant de les garder pour les questions suivantes.

Il y a 4 faux-positifs au total.

6.2) (3 points) Donnez la valeur des sous-séquences correspondants aux faux positifs que vous avez trouvés dans 6.1.

63, 74, 41, 63

6.3) (2 points) Rabin-Karp produit parfois des faux-positifs (candidats retenus ne remplissant pas le critère de correspondance). Expliquez pourquoi il ne génère pas de faux-négatifs (candidats rejetés remplissant le critère de correspondance).

Si un candidat est valide, la valeur de hash qu'il produira est forcément la bonne et il est donc impossible de rejeter un candidat valide.

6.4) On veut rechercher deux patrons P_1 et P_2 de longueur *m* dans un texte T de longueur *n*, et ce en exécutant une variante de l'algorithme Rabin-Karp (appliqué à deux patrons de même longueur et non plus un).

6.4.1) (2 points) Quelle est la complexité (en pire cas) de cet algorithme ? Justifiez brièvement.

Le pire cas survient si chaque décalage signale un candidat potentiel et que les deux patrons P_1 et P_2 ont la même valeur de hash car il faudrait alors effectuer deux comparaisons sur m caractères chaque fois qu'un candidat est trouvé. Il en résulte alors que la complexité en pire cas est $O(2m(n-m+1)) = O(m(n-m+1))$.

6.4.2) (2 points) Combien de faux-positifs produit cet algorithme sur la chaîne de caractères de la question 6.1 (96325874124896323495) en recherchant les patrons 23 et 87 ?

Nous aurons 3 faux-positifs pour 23 et 2 feux-positifs pour 87, donc un total de 5 faux-positifs.

6.4.3) (2 points) Combien de faux-négatifs produit cet algorithme en recherchant les mêmes patrons ?

Aucun. Rabin-Karp ne produit pas de faux-négatifs.

6.4.4) (2 points) Quels sont les décalages retournés pour chacun de ces patrons ?

Pour 23, nous aurons le décalage 15.

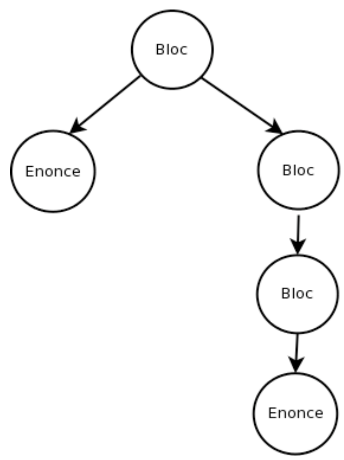
Pour 87, nous aurons le décalage 5.

Question 7 : Métriques dans les arbres de syntaxe abstraits (10 points)

Un AST est une représentation abstraite de la syntaxe du code d'une application. Pour simplifier le problème, on ne représente ici que les blocs et les énoncés dans le code. Les blocs peuvent contenir d'autres blocs ou des énoncés, mais les énoncés ne peuvent avoir aucun enfant. Par exemple, le code:

```
public int uneMethode() {
   int i = 0;
   while(i < 12) {
      if(i > 4) {
         System.out.println(i);
      }
   }
}
```

Serait représenté par l'AST suivant:



On désir connaître le nombre d'énoncés et de blocs contenus à l'intérieur de chacun des blocs de l'AST. Par exemple, le bloc racine (qui correspond au bloc `uneMethode`) contient 2 blocs et 2 énoncés. Proposez un algorithme en temps linéaire qui permet de calculer les métriques de chacun des blocs de l'AST. <u>*Indice:*</u> *Il faut parcourir tous les nœuds du graphe pour faire ce calcul.*

Il suffit de faire une fouille en profondeur et d'accumuler les métriques en les faisant remonter progressivement vers leur parent.

Question 8 : Programmation dynamique (10 points)

Les coefficients binomiaux ont une grande utilité en mathématique puisqu'on les retrouve par exemple dans le calcul des combinatoires ou le binôme de Newton. La combinatoire *k* de *n* ($k \leq n$) est donnée par :

$$\binom{n}{k} = C_n^k = \frac{n!}{k!(n-k)!}$$

Il est également possible de l'exprimer par une formule de récurrence :

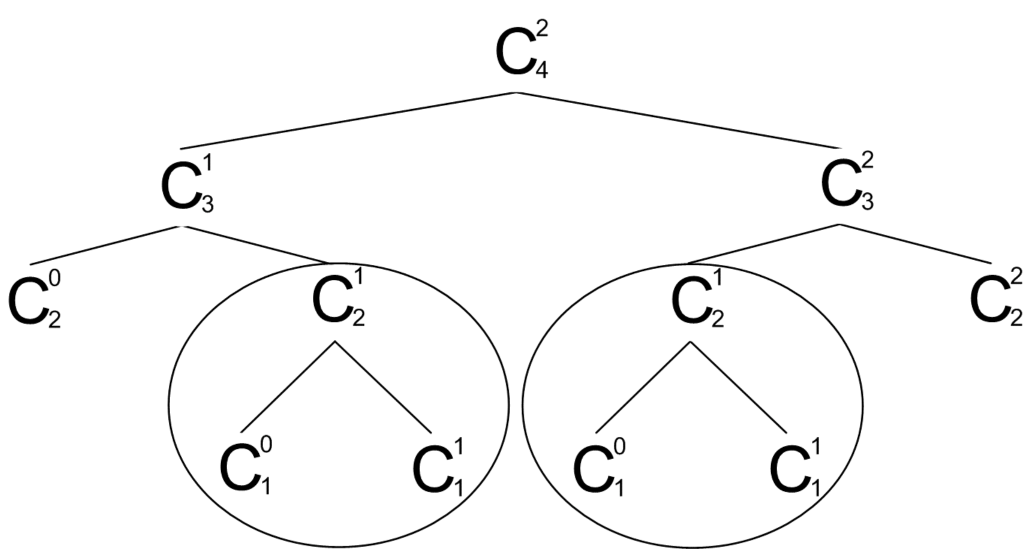
$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$$

où $C_0^0 = C_n^0 = C_n^n = 1$. Cela peut se traduire par l'implémentation suivante :

```
public static int Combinatoire(int n, int k)
{
	if(n <0 || k<0)
		return -1;

	if(n==k || k==0)
		return 1;
	else
		return Combinatoire(n-1, k) +  Combinatoire(n-1, k-1);
}
```

8.1) (3 points) En vous appuyant sur la fonction donné ci-dessus, illustrez le fait que l'expression récursive des coefficients binomiaux présente un cas de recouvrement des sous-problèmes en vous basant sur les étapes de calcul de C_4^2.



8.2) On vous propose l'implémentation d'un algorithme de programmation dynamique suivante pour le calcul des coefficients binomiaux :

```
public static int CombinatoireDynamique(int n, int k)
{
      if(n <0 || k<0)
            return -1;

      int[][] M = new int[ n+1 ][ k+1 ];

      for(int i=0; i<=n; i++)
      {
            for(int j=0; j <= (i<k ? i : k) ; j++)
            {
                  if(i==j || j==0)
                        M[ i ][ j ] = 1;
                  else
                        M[ i ][ j ] = M[ i-1 ][ j ] +  M[ i-1 ][ j-1 ];
            }
      }

      return M[ n ][ k ];
}
```

8.2.1) (3 points) Détaillez les étapes d'exécution de cet algorithme pour le calcul de C_4^2

1				
1	**1**			
1	**2**	**1**		
1	**3**	**3**		
1	**4**	**<u>6</u>**		

8.2.2) (2 points) Peut-on réduire l'espace mémoire nécessaire pour le tableau M[n+1][k+1] ? Justifiez votre réponse.

Oui, puisque la ligne i ne dépend que de la ligne i-1, les calculs des lignes antérieures ne sont plus nécessaires pour la suite des opérations.

8.2.3) (2 points) Quelle est selon vous la plus petite taille (en fonction des paramètres n et k) de la table M[][] qui permette l'implémentation d'un algorithme de programmation dynamique pour le calcul des coefficients binomiaux ? Justifiez.

Il est possible de réduire l'espace mémoire à 1×k. Il faut cependant changer le parcours des cases (de droite à gauche et non plus de gauche à droite). La réponse 2×k est acceptée en cela qu'elle illustre clairement la compréhension de la question 8.2.2.

Annexe I

Nom et prénom :___________________________

Matricule : _______________________

	Arcs incidents							
A								
B								
C								
D								
E								
F								
G								
H								
Entrée								
Sortie								

	Arcs incidents							
A								
B								
C								
D								
E								
F								
G								
H								
Entrée								
Sortie								