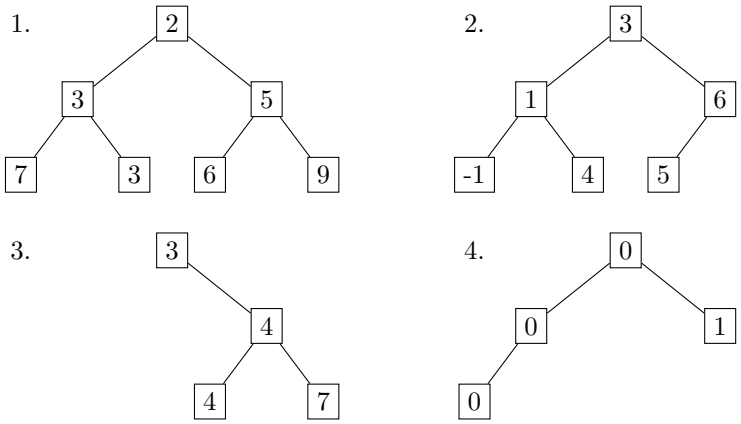
Consignes générales : le sujet se compose de deux parties indépendantes. Il est permis de rédiger les deux problèmes sur une même copie. Le langage de programmation naturel pour cet examen est le C++, mais n'est pas obligatoire, à condition que l'étudiant rédige dans un (méta)langage précis et consistant. Il est toujours possible de faire appel à un algorithme auxiliaire s'il est préalablement défini dans la copie.

Partie 1 : Reconnaissance d'Arbres Binaires Structurés (12pts)

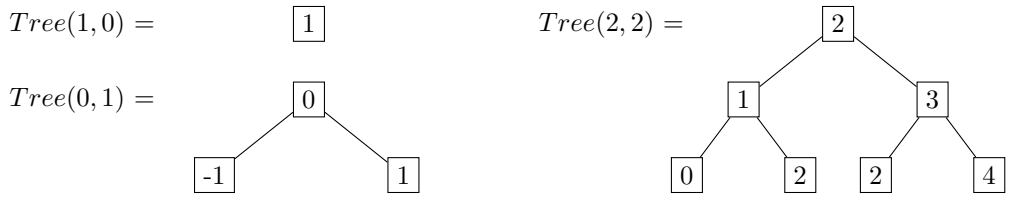
Dans cette section, les sommets d'un arbre sont étiquettés par des entiers quelconques. Deux sommets peuvent avoir la même étiquette. Les définitions d'arbre parfait, de tas et d'arbre binaire de recherche sont celles du cours et ne sont donc pas redonnées. Toutefois, nous rappelons que les tas considérés sont de type *min heap*, c'est à dire que la clef minimale se trouve à la racine.

Analyse d'exemples (5pts)

Question 1 (2pts)**.** *Déterminer lesquels des quatre arbres binaires suivants sont des arbres parfaits, des tas, ou des arbres binaires de recherche (attention, un même arbre peut appartenir à plusieurs classes).*



Soit x un entier et h un entier positif, on définit l'arbre $Tree(x, h)$ par récurrence sur h comme suit: $Tree(x, 0)$ est une feuille dont l'étiquette a valeur x. $Tree(x, h)$ est un arbre de hauteur h ayant pour racine x, pour sous arbre gauche $Tree(x\text{-}1, h\text{-}1)$ et pour sous arbre droit $Tree(x\text{+}1, h\text{-}1)$. Par exemple:



Question 2 (2pts)**.** *Existe-t-il des valeurs de x et h telles que $Tree(x, h)$ ne soit pas :*
1. un arbre parfait ? 2. un tas ? 3. un arbre binaire de recherche ? (justifier)
Indication: lorsque de telles valeurs existent, il est suffisant de donner un exemple pour se justifier, sinon il faut expliquer pourquoi il est impossible d'en trouver.

Question 3 (1pt)**.** *Le nombre de sommets de $Tree(x, h)$ dépend-il de x, de h ou des deux variables ? Calculer le nombre de sommets de $Tree(x, h)$ en fonction des paramètres choisis.*

Algorithmes de reconnaissance (7pts)

On définit l'indice d'un sommet s, noté $Index(s)$, par induction structurelle sur la hauteur d'un arbre : si s est une racine, $Index(s) = 1$, sinon si s est le fils gauche de r, $Index(s) = 2 \times Index(r)$, et sinon, quand s est le fils droit de r, $Index(s) = 2 \times Index(r) + 1$. L'indice maximum d'un arbre binaire T, dénoté $Index(T)$, est l'indice le plus grand de ses sommets. Par convention, un arbre vide a pour indice maximum 0. Enfin, le nombre de sommets d'un arbre T est dénoté par $Size(T)$.

Question 4 (1pt)**.** *Ecrire un algorithme qui calcule l'indice maximum d'un arbre binaire.*

Question 5 (1pt)**.** *Justifier qu'un arbre binaire T est parfait si et seulement si $Index(T) = Size(T)$.* Indication: observer les indices des sommets dans un parcours en largeur gauche-droite.

Question 6 (1pt)**.** *En déduire un algorithme qui teste si un arbre binaire est parfait.*

Question 7 (2pts)**.** *Ecrire un algorithme qui teste si un arbre binaire est un tas. Préciser sa complexité.*

Question 8 (2pts)**.** *Ecrire un algorithme linéaire qui teste si un arbre binaire est de recherche.* Indication: si besoin, il est possible d'utiliser les fonctions "int IntMin()" et "int IntMax()" qui renvoient respectivement le plus petit et le plus grand entier encodable dans un int.

Partie 2 : Jeu de 52 Cartes (8pts)

Dans cette section, on s'intéresse à la structure *carte*{*int enseigne, int valeur*}. Une carte d'enseigne e et de valeur v est aussi dénotée par (e, v). Par convention, on a :

$e = 0$ représente les coeurs ♡,	$v = 1$ représente les as,
$e = 1$ représente les piques ♠,	$v = 11$ représente les valets,
$e = 2$ représente les carreaux ♢,	$v = 12$ représente les dames,
$e = 3$ représente les trèfles ♣.	$v = 13$ représente les rois.

$1 < v \leqslant 10$ représente les autres valeurs possibles.

Nous disons qu'une carte est noire si son enseigne est pique ♠ ou trèfle ♣, et nous disons qu'une carte est rouge si elle est coeur ♡ ou carreau ♢. Par exemple, la carte $(2, 1)$ représente l'as de carreau qui est une carte rouge et l'ensemble $[(3, 12); (1, 12)]$ est la paire de dames noires.

Question 9 (1pt)**.** *Indiquer quelles sont les cartes présentes dans l'ensemble* $[(0, 3); (3, 1); (1, 13); (2, 4)]$.

Question 10 (2pts)**.** *Ecrire un algorithme optimal qui détermine si toutes les cartes d'un ensemble sont de la même enseigne, et un second algorithme optimal qui détermine si un ensemble contient au moins une carte de chaque couleur. Justifier que leur complexité est optimale dans le pire des cas.*

On souhaite trier des cartes dans l'ordre lexicographique, c'est à dire en regroupant d'abord les enseignes, puis en triant les enseignes par valeur croissante. Formellement, la carte (e_1, v_1) se situe avant la carte (e_2, v_2) dans ce tri si et seulement si $e_1 < e_2$ ou $e_1 = e_2$ et $v_1 \leqslant v_2$. Ainsi, l'as de pique $(1, 1)$ se situe avant la dame de pique $(1, 12)$ qui se trouve avant le trois de trèfle $(3, 3)$.

Question 11 (2pts)**.** *Sachant qu'il y a un nombre constant de cartes différentes, donner un algorithme linéaire pour trier un ensemble de carte. (indication: utiliser le tri panier, ou tri par comptage)*

Question 12 (1pt)**.** *Pourquoi est-il possible d'implémenter un algorithme qui trie un paquet de cartes contenant exactement un exemplaire de chaque carte en temps constant ?*

Question 13 (2pts)**.** *Si l'on suppose qu'il n'y a plus de limite sur le nombre d'enseignes ou sur le nombre de valeurs, que devient la complexité d'un tri de cartes par ordre lexicographique ?*