main grâce à srand(time(NULL)).

TP2: Tas

L'objectif de ce TP est d'implanter les opérations de base sur les tas afin d'arriver à l'algorithme du tri par tas.

Fichiers, tests, etc.

La fonction main est contenue dans le fichier tests.cc, et la compilation est gérée par le Makefile. Le fichier d'en-têtes Tas.h est complet. Le seul fichier à compléter est le fichier Tas.cc.

Il est impératif de tester toutes vos fonctions et de bien vérifier que le résultat attendu est le bon! Pour tester, les tableaux manipulés sont affichés comme arbres quasi-complet dans le fichier arbre. svg. L'affichage sous forme de tableau n'est pas fourni, et est l'objet de la première question.

Le fichier test fournit deux tableaux prédéfinis : le *tableau test* [7,12,4,8,10,2,1,1,3,9] (qui n'est ni un tas max ni un tas min) et le *tas test* [9,7,8,6,4,0,2,3,5,1] (qui est un tas max). Leurs représentations sous forme d'arbre sont fournies en figure 1.

Exercice à rendre

À part dans la question 2, le terme tas signifie toujours tas max dans cet exercice.

- 1. Compléter la fonction void afficher (int n, int* T) qui affiche le tableau T de taille n, sous la forme suivante : « [7,12,4,8,10,2,1,1,3,9] » (entre crochet, avec des virgules pour séparer, sans espace).
- 2. Compléter les fonctions bool estTasMax(int n, int* T) et bool estTasMin(int n, int* T) qui testent si T est un tas max ou un tas min, respectivement.
- **3.** Compléter la fonction void tableauManuel(int n, int* T) qui remplit le tableau T de taille n par des valeurs rentrées à la main par l'utilisateur. On utilisera cin pour rentrer des valeurs à la main.
- 4. Compléter la fonction void tableauAleatoire(int n, int* T, int m, int M) qui remplit le tableau T avec n entiers aléatoires compris entre m et M (inclus).
 Utiliser rand()%k qui renvoie un entier aléatoire entre 0 et k-1. La graine est initialisée dans la fonction
- 5. Compléter fonction void entasser(int n, int* T, int i) qui entasse le nœud d'indice i dans le tableau T de taille n.
- **6.** Utiliser la fonction entasser pour compléter la fonction void tas(int n, int* T) qui transforme le tableau *T* de taille *n* en un tas.
- 7. Compléter la fonction int* trier(int n, int* T) qui implémente l'algorithme TRITAS du cours et renvoie un tableau qui contient les valeurs de T triées en ordre croissant.
- **8.** Réaliser une implémentation **TrierSurPlace** de TRITAS qui trie le tableau sur place, c'est-à-dire sans utilisation de tableau annexe.

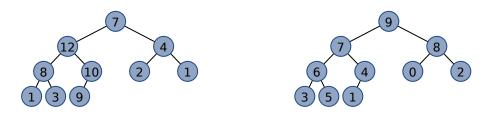


FIGURE 1 – Tableau test (à gauche) et tas test (à droite)