Récupérez les fichiers `fichierTP4.cpp`, `progListeSC.h`, `progListeSC.cpp`.
Le fichier `fichierTP4.cpp` contient le `main` et la définition du type `ArbreBin` avec la fonction `creerArbreBin` et des fonctions pour l'affichage d'un arbre. Il contient les entêtes d'autres fonctions sur les arbres que vous aurez à compléter.
Les fichiers `progListeSC.h` et `progListeSC.cpp` contiennent la définition du type `ListeSC`.
L'affichage des arbres utilisera les langages et outils `dot` du logiciel `Graphviz` pour la visualisation des graphes.
Vous compilerez avec le commande `g++ -Wall -ansi -pedantic fichierTP4.cpp progListeSC.cpp -o tp4`

1. Complétez la fonction `sommeNoeuds` qui renvoie la somme des valeurs des noeuds d'un arbre binaire.
   Complétez la fonction `profMinFeuille` qui renvoie la profondeur minimum des feuilles d'un arbre binaire.
   Exemple : pour l'arbre `A` de la figure 1, `sommeNoeuds(A)` renvoie 27, `profMinFeuille(A)` renvoie 2.

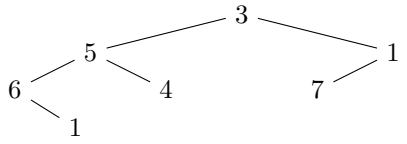
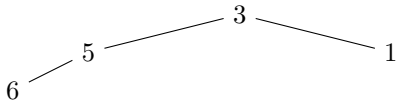


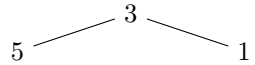
FIGURE 1 – un arbre binaire A

   Compilez et exécutez `tp4` avec l'option 1 pour tester vos fonctions.
2. Complétez la fonction `parcoursInfixe` qui renvoie la liste des valeurs des noeuds de l'arbre dans l'ordre infixe. Vous utiliserez la fonction de concaténation de listes `concatLSC` (voir ses spécifications dans `progListeSC.h`).
   Exemple : avec l'arbre de la figure 1 `parcoursInfixe(A)` renvoie la liste `(6,1,5,4,3,7,1)`.
   Compilez et exécutez `tp4` avec l'option 2 pour tester votre fonction.
3. Complétez la fonction `effeuiller` qui modifie l'arbre en supprimant ses feuilles.
   Exemple : après l'exécution de `effeuiller(A)` avec l'arbre de la figure 1, l'arbre `A` devient



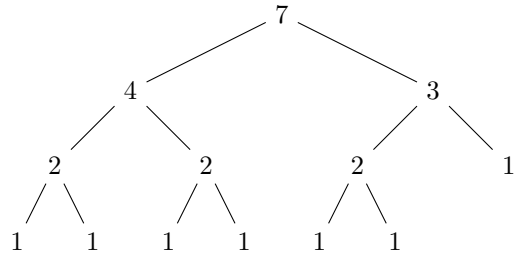
l'arbre binaire A effeuillé

   Compilez et exécutez `tp4` avec l'option 3 pour tester votre fonction.
4. Complétez la fonction `tailler` qui étant donné un arbre binaire `A` et un entier positif `p` modifie l'arbre A, en supprimant les noeuds de profondeur au moins p.
   Exemple : après l'exécution de `tailler(A,2)` avec l'arbre de la figure 1, l'arbre `A` devient



l'arbre binaire A taillé à la profondeur 2

   Compilez et exécutez `tp4` avec l'option 4 pour tester votre fonction.
5. Complétez la fonction `genererAB` qui étant donné un entier $n > 0$, renvoie l'arbre binaire, dont la racine est n et dont chaque noeud de valeur $k > 1$ a 2 fils, le fils gauche de valeur $k - k/2$, le fils droit de valeur $k/2$. Les noeuds de valeur 1 sont les feuilles de l'arbre.



   Exemple : `genererAB(7)` renvoie l'arbre
   Compilez et exécutez `tp4` avec l'option 5 pour tester votre fonction.

6. Complétez la fonction `tronconner` qui supprime les noeuds ayant exactement un fils d'un arbre binaire.

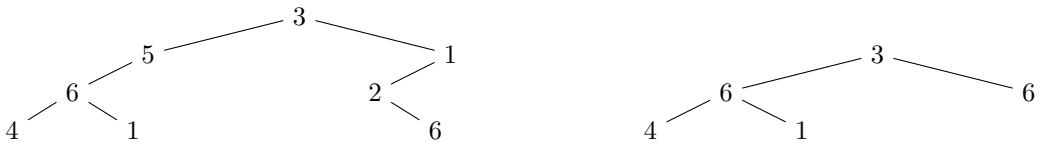


FIGURE 2 – à gauche un arbre binaire ; à droite l'arbre binaire après tronçonnage.

Compilez et exécutez `tp4` avec l'option 6 pour tester votre fonction.

7. Complétez la fonction `estParfait(A)` qui teste si l'arbre binaire `A` est un arbre parfait :
pour tout entier naturel i inférieur ou égal à la hauteur de A, le nombre de noeuds de A de profondeur i est 2^i.
Exemple :

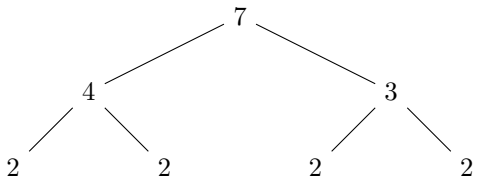


FIGURE 3 – arbre parfait

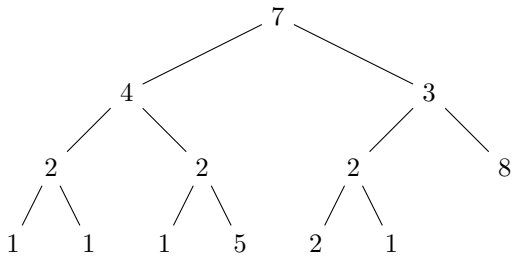


FIGURE 4 – arbre non parfait

Testez votre fonction avec l'option 7.