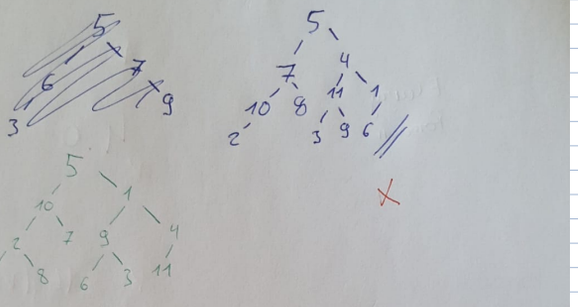
Parcours d'arbres [4 points]

a. Dessinez l'arbre **binaire** dont on connaît les résultats des parcours suivants :

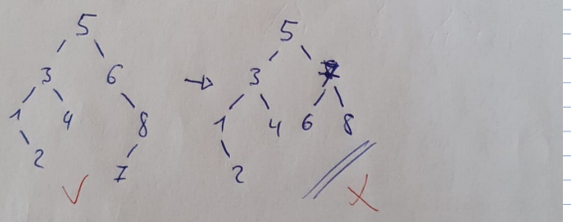
Parcours post-ordonné : 8 - 2 - 7 - 10 - 6 - 3 - 9 - 11 - 4 - 1 - 5

Parcours symétrique : 2 - 8 - 10 - 7 - 6 - 6 - 9 - 3 - 1 - 11 - 4



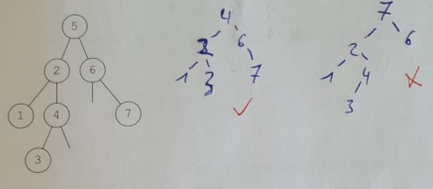
b. Dessinez l'arbre **binaire de recherche** dont le parcours post-ordonné est le suivant :

2 – 1 – 4 – 3 – 7 – 8 – 6 – 5

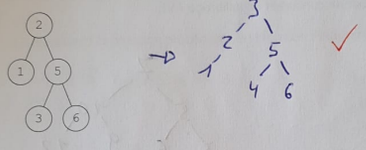


2. Arbres binaires de recherche et arbres AVL [10 points]

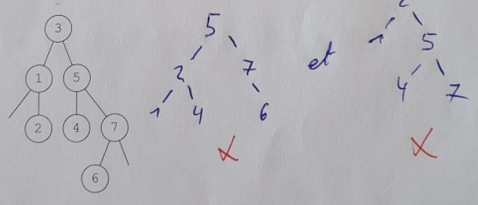
a. Soit **l'arbre binaire de recherche** suivant, dessinez les 2 arbres pouvant résulter de la suppression de sa racine. Ne faites pas d'équilibrage AVL.



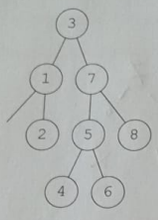
b. Soit **l'arbre AVL** suivant, dessinez l'arbre résultant de l'insertion de la clé 4, après rotations éventuelles pour en conserver l'équilibrage AVL

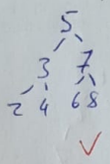


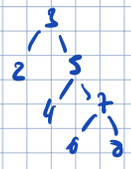
c. Soit **l'arbre AVL** suivant, dessinez l'arbre résultant de la suppression de la clé 3, après rotations éventuelles pour en conserver l'équilibrage AVL. Donnez les deux solutions possibles.

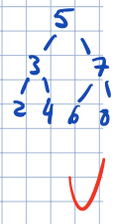


d. Soit **l'arbre AVL** suivant, dessinez l'arbre résultant de la suppression de la clé 1 après rotations éventuelles pour en conserver l'équilibrage AVL

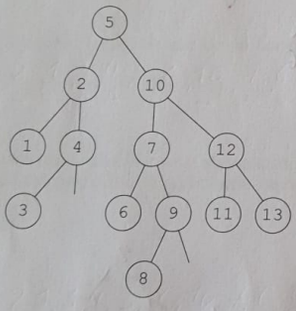


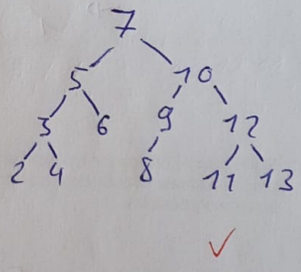


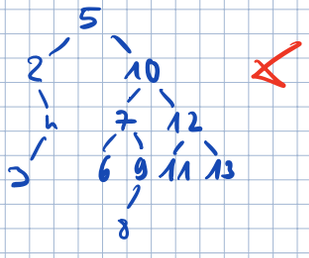




e. Soit **l'arbre AVL** suivant, dessinez l'arbre résultant de la suppression de la clé 1 après rotations éventuelles pour en conserver l'équilibrage AVL.







HEIG-VD – Département TIC ASD TE2 01.06.2023

Nom : Prénom :

3. Expressions arithmétiques [4 pts] 3,5

a. On applique l'algorithme de Dijkstra à deux piles à l'expression arithmétique suivante :

```
(1 * ((2 + 3) - (4 / (5 + 6))))
```

- Quel est l'état de la pile des valeurs après l'insertion du 6 ? (Écrire la pile avec le sommet à droite)

1 5 4 5 6 ✓

- Quel est l'état de la pile des opérateurs après l'insertion du 6 ? (Écrire la pile avec le sommet à droite)

* - / + ✓

b. Récrivez l'expression précédente en Notation Polonaise Inverse (NPI) :

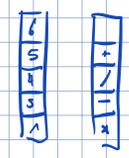
1 2 3 + 4 5 6 + / - * ✓

c. Réécrivez l'expression Polonaise Inverse (NPI) suivante en notation infixe (avec tout les parenthèses).

1 2 + 3 4 * / 5 6 7 + / -

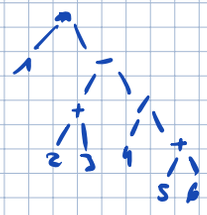
((1+2)/(3*4)) - (5/(6+7))

Profs Guy-Michel Breguet, Olivier Cuisenaire, Laura Elena Raileanu 5/12



1 5 4 5 6 ✓

* - / +



NPI : 1 2 3 + 4 5 6 + / - * ✓

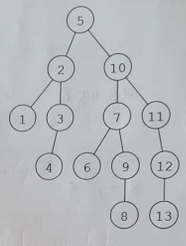
((((1+2) / (3*4)) - (5/(6+7)))) ✓

HEIG-VD – Département TIC ASD TE2 01.06.2023

4. Itération sur un arbre [4 pts] 4

Soit l'arbre et la structure Nœud suivants. Tracez les nœuds par lesquels passe l'itération et soulignez les nœuds visités en pré-ordre et en post-ordre.

```
structure Noeud<T> {
    T etiquette
    Noeud* aine
    Noeud* puine
    Noeud* parent
}
```



a. Pré-ordre :

5, 2, 1, 3, 4, 3, 2, 10, 7, 6, 9, 8, 9, 7, 11, 12, 13, 12, 11, 10, 5

✓✓

5 2 1 3 4 3 2 10 7 6 9 8 9 7 11 12 13 12 11 10 5 ✓

b. Post-ordre :

5, 2, 1, 3, 4, 3, 2, 10, 7, 6, 9, 8, 9, 7, 11, 12, 13, 12, 11, 10, 5

✓✓

5 2 1 3 4 3 2 10 7 6 9 8 9 7 11 12 13 12 11 10 5 ✓

Profs Guy-Michel Breguet, Olivier Cuisenaire, Laura Elena Raileanu 6/12

HEIG-VD – Département TIC ASD TE2 01.06.2023

Nom : Prénom : 3

5. Tas [5 pts]

Effectuez le tri par tas sur le tableau de 8 éléments suivant en effectuant en premier make_heap puis sort_heap :

1	8	3	4	5	6	7	9

Indiquez l'état du tableau après chaque descente d'éléments, ainsi que le résultat final. Il est possible qu'il y ait plus de lignes que nécessaire dans les tableaux ci-dessous.

a. make_heap :

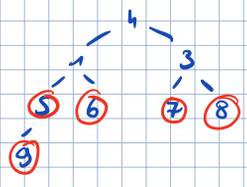
8	1	3	4	5	6	7	9
8	1	3	4	5	6	7	9
8	4	3	1	5	6	7	9
8	5	3	1	4	6	7	9
8	5	6	1	4	3	7	9
8	5	7	1	4	3	6	9
9	8	7	5	4	3	6	1

b. sort_heap :

8	5	7	1	4	3	6	9
7	5	6	1	4	3	8	9
6	5	3	1	4	7	8	9
5	4	3	1	6	7	8	9
4	1	3	5	6	7	8	9
3	1	4	5	6	7	8	9
1	3	4	5	6	7	8	9

Profs Guy-Michel Breguet, Olivier Cuisenaire, Laura Elena Raileanu 7/12

1	8	3	4	5	6	7	9
1	8	3	9	5	6	7	4
1	9	3	8	5	6	7	4
1	8	7	8	5	6	3	4
9	1	7	8	5	6	3	4
9	8	7	1	5	6	3	4
9	8	7	4	5	6	3	1



9	8	7	4	5	6	3	1
1	8	7	4	5	6	3	9
8	5	7	4	1	6	3	9
3	5	7	4	1	6	8	9
7	5	6	4	1	3	8	9
3	5	6	4	1	7	8	9
6	5	3	4	1	7	8	9
1	5	3	4	6	7	8	9
5	4	3	1	6	7	8	9
1	4	3	5	6	7	8	9
4	1	3	5	6	7	8	9
3	1	4	5	6	7	8	9
1	3	4	5	6	7	8	9

HEIG-VD – Département TIC ASD TE2 01.06.2023

6. Feuilles d'un binaire [5 pts] 4

Étant donné un arbre binaire dont les nœuds utilisent la structure C++ suivante

```
struct Noeud {
    string etiquette;
    Noeud* parent;
    Noeud* gauche;
    Noeud* droite;
};
```

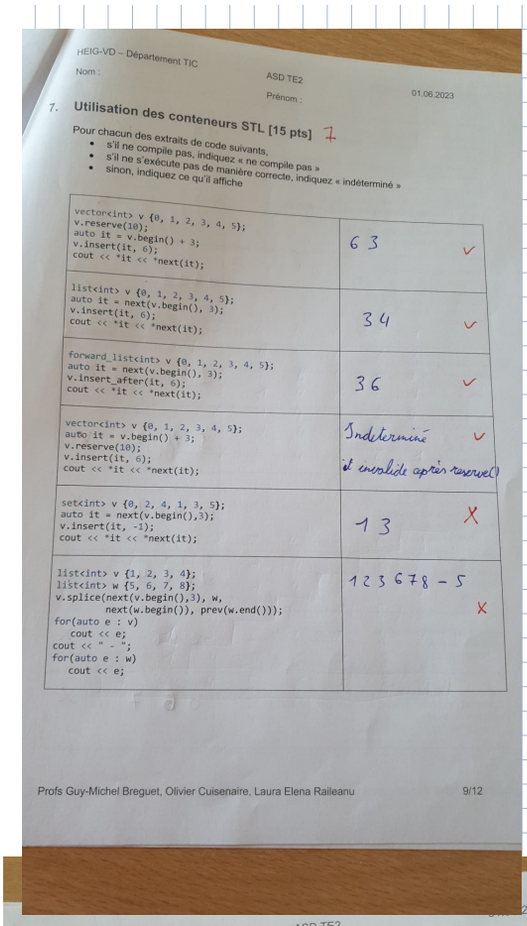
Écrivez une fonction qui affiche en pré-ordre toutes les feuilles de cet arbre binaire à la sortie standard, et dont le prototype est

```
void afficher_feuilles(const Noeud* n);
```

mes nullptr? X

```
if (n.gauche == nullptr && n.droit == nullptr){
    std::cout << n etiquette;
} else {
    afficher_feuilles(n.gauche);
    afficher_feuilles(n.droite);
}
```

rofs Guy-Michel Breguet, Olivier Cuisenaire, Laura Elena Raileanu 8/12



HEIG-VD – Département TIC

Nom :

ASD TE2

Prénom :

01.06.2023

7. **Utilisation des conteneurs STL [15 pts]** 7

Pour chacun des extraits de code suivants,
- s'il ne compile pas, indiquez « ne compile pas »
- s'il ne s'exécute pas de manière correcte, indiquez « indéterminé »
- sinon, indiquez ce qu'il affiche

`vector<int> v {0, 1, 2, 3, 4, 5};`
`v.reserve(10);`
`auto it = v.begin() + 3;`
`v.insert(it, 6);`
`cout << *it << *next(it);`
	6 3 ✓

`list<int> v {0, 1, 2, 3, 4, 5};`
`auto it = next(v.begin(), 3);`
`v.insert(it, 6);`
`cout << *it << *next(it);`
	3 4 ✓

`forward_list<int> v {0, 1, 2, 3, 4, 5};`
`auto it = next(v.begin(), 3);`
`v.insert_after(it, 6);`
`cout << *it << *next(it);`
	3 6 ✓

`vector<int> v {0, 1, 2, 3, 4, 5};`
`auto it = v.begin() + 3;`
`v.reserve(10);`
`v.insert(it, 6);`
`cout << *it << *next(it);`
	Indeterminé ✓
	it envalide après reserve()

`set<int> v {0, 2, 4, 1, 3, 5};`
`auto it = next(v.begin(),3);`
`v.insert(it, -1);`
`cout << *it << *next(it);`
	1 3 ✗

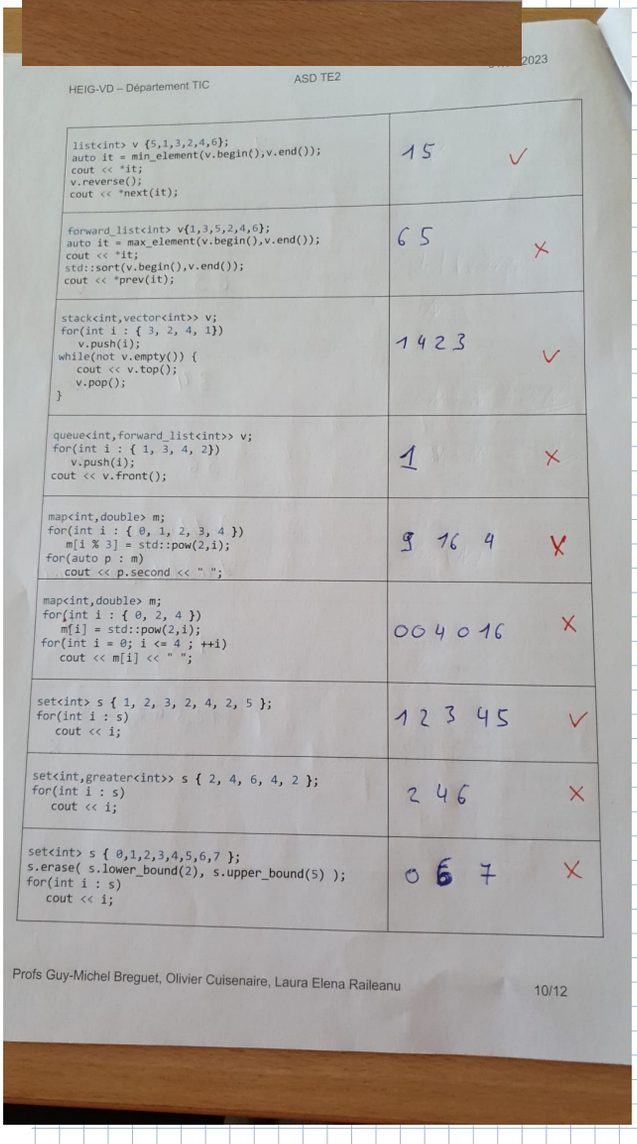
`list<int> v {1, 2, 3, 4};`
`list<int> w {5, 6, 7, 8};`
`v.splice(next(v.begin(),3), w,`
`next(w.begin()), prev(w.end()));`
`for(auto e : v)`
`cout << e;`
`cout << " - ";`
`for(auto e : w)`
`cout << e;`
	1 2 3 6 7 8 – 5 ✗

Profs Guy-Michel Breguet, Olivier Cuisenaire, Laura Elena Raileanu 9/12

63

34

Vector
list
forward list
set
stack
queue
map



HEIG-VD – Département TIC

ASD TE2

2023

`list<int> v {5,1,3,2,4,6};`
`auto it = min_element(v.begin(),v.end());`
`cout << *it;`
`v.reverse();`
`cout << *next(it);`
	1 5 ✓

`forward_list<int> v{1,3,5,2,4,6};`
`auto it = max_element(v.begin(),v.end());`
`cout << *it;`
`std::sort(v.begin(),v.end());`
`cout << *prev(it);`
	6 5 ✗

`stack<int,vector<int>> v;`
`for(int i : { 3, 2, 4, 1})`
`v.push(i);`
`while(not v.empty()) {`
`cout << v.top();`
`v.pop();`
`}`
	1 4 2 3 ✓

`queue<int,forward_list<int>> v;`
`for(int i : { 1, 3, 4, 2})`
`v.push(i);`
`cout << v.front();`
	1 ✗

`map<int,double> m;`
`for(int i : { 0, 1, 2, 3, 4 })`
`m[i % 3] = std::pow(2,i);`
`for(auto p : m)`
`cout << p.second << " ";`
	9 16 4 ✗

`map<int,double> m;`
`for(int i : { 0, 2, 4 })`
`m[i] = std::pow(2,i);`
`for(int i = 0; i <= 4 ; ++i)`
`cout << m[i] << " ";`
	0 0 4 0 16 ✗

`set<int> s { 1, 2, 3, 2, 4, 2, 5 };`
`for(int i : s)`
`cout << i;`
	1 2 3 4 5 ✓

`set<int,greater<int>> s { 2, 4, 6, 4, 2 };`
`for(int i : s)`
`cout << i;`
	2 4 6 ✗

`set<int> s { 0,1,2,3,4,5,6,7 };`
`s.erase( s.lower_bound(2), s.upper_bound(5) );`
`for(int i : s)`
`cout << i;`
	0 6 7 ✗

Profs Guy-Michel Breguet, Olivier Cuisenaire, Laura Elena Raileanu 10/12

Nom : Prénom :

8. Complexités [10 pts] 4

Quelle est la complexité (dans le pire cas si cela dépend des données) des fonctions suivantes en fonction des variables N (et M, si applicable) ?

```
vector<int> f1(size_t N) {
   vector<int> v;
   for(size_t i = 0; i < N; ++i)
      v.insert(v.begin(),int(i));
   return v;
}
```

N^2 N^2 ✓

```
list<int> f2(size_t N) {
   list<int> v;
   for(size_t i = 0; i < N; ++i)
      v.insert(v.begin(),int(i));
   return v;
}
```

N N ✓

```
set<int> f3(size_t N) {
   set<int> s;
   for(size_t i = 0; i < N; ++i)
      s.insert(rand());
   return s;
}
```

$N \cdot Log(Log(N))$ $N\log(N)$ ✗

```
vector<int> f4(size_t N) {
   vector<int> v;
   for(size_t i = 0; i < N; ++i)
      v.insert(next(v.begin(),v.size()),int(i));
   return v;
}
```

N N

N N^2 ✓

```
list<int> f5(size_t N) {
   list<int> v;
   for(size_t i = 0; i < N; ++i)
      v.insert(next(v.begin(),v.size()),int(i));
   return v;
}
```

N N ✗

```
set<bool> f6(int N) {
   set<bool> s;
   for(int i = 0; i < N; ++i)
      s.insert(rand() % 2 == 0);
   return s;
}
```

$N \log(N)$ ✗


```
vector<int> f7(size_t N) {
   vector<int> v(N);
   generate(v.begin(),v.end(),rand);
   make_heap(v.begin(),v.end());
   return v;
}
```

$2N$ N^2 ✓

```
list<int> f8(list<int>& v) {
   size_t N = v.size();
   list<int> w;
   w.splice(w.begin(),v,v.begin(),v.end());
   return w;
}
```

1

N ✗

```
stack<vector<int>> f9(size_t N, size_t M) {
   stack<vector<int>> s;
   for(size_t i = 1; i < N; i *= 2) {
      s.emplace(M);
      for (size_t j = 0; j < M; ++j) {
         s.top().at(j) = rand();
      }
      sort(s.top().begin(),s.top().end());
   }
   return s;
}
```

$2N + N^2 \Rightarrow N^2$ ✗

$N^2 \log(M)$

```
map<int,size_t> f10(vector<int> const& v) {
   size_t N = v.size();
   map<int,size_t> m;
   for(size_t i = 0; i < N; ++i)
      m[v[i]] = i;
   return m;
}
```

$\log(N)$ ✗

$N^2 \log(N)$