Télécharger exercices-arbre-binaire.zip sur le site https://cviroulaud.github.io

Exercice 1 : L'arbre figure 1 représente une opération arithmétique.

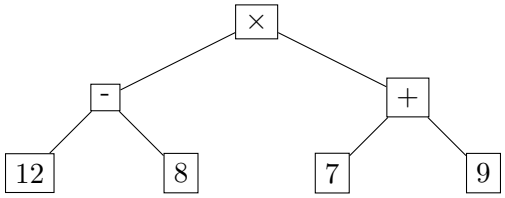

FIGURE 1 – Calcul arithmétique

1. Parcourir cet arbre en profondeur (préfixe, infixe et postfixe).
2. Donner le résultat du calcul.
3. Pour quel parcours est-il indispensable de rajouter des parenthèses ?

Exercice 2 :

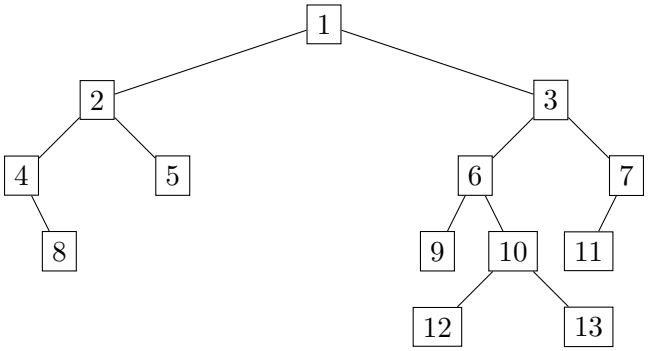

FIGURE 2 – Arbre binaire

1. Effectuer les parcours suivants dans l'arbre figure 2 :
   - en largeur,
   - préfixe,
   - infixe,
   - postfixe.
2. Quelle est la hauteur de cet arbre. On considère qu'un arbre vide à une hauteur de -1.
3. Cet arbre est-il équilibré ?
4. Cet arbre est-il complet ?

Exercice 3 : En généalogie le principe de la numérotation de Sosa-Stradonitz attribue le numéro 1 à l'individu racine (le sujet sur lequel on établit l'ascendance) puis le numéro 2 à son père et 3 à sa mère. Ainsi, chaque homme a un numéro double de celui de son enfant et est donc pair et chaque femme un numéro double de celui de son enfant plus un, soit un numéro impair.

1. Le numéro 17 est-il une femme ou un homme ? Quels sont les numéros de son père et sa mère ? Quel est le numéro de son enfant ?
2. Combien de trisaïeuls (ancêtres de quatrième degré) possède un individu ?
3. Un généalogiste propose d'imprimer l'arbre généalogique d'un individu avec quatre degrés d'ascendance. Combien de personnes seront affichées dans cet arbre ?

L'arbre des ascendants d'Emmanuel Macron est représenté dans un liste Python (code 1).

Christophe Viroulaud

```
# Pour respecter la numérotation de Sosa-Stradonitz la première cellule est laissée vide
macron = ["", "Emmanuel Macron", "Jean-Michel Macron", "Françoise Noguès",
        "André-Henri Macron", "Jacqueline Robertson", "Jean Noguès",
        "Germaine Arribet", "Henri Eugène Macron", "Marie Adèle Bosseur",
        "Georges William Robertson", "Suzanne Leblond", "Fabien Noguès",
        "Esther Mas", "Ernest Arribet", "Marie Madeleine Millet"]
```

Code 1 – Représentation en mémoire d'un arbre généalogique

4. Représenter l'arbre généalogique sous forme d'un arbre binaire.
5. Ouvrir le fichier *arbre-genealogique.py*.
6. Écrire la fonction **get_parents(tab : list, id_enfant : int) → tuple** qui renvoie le nom des parents de *id_enfant* sous forme de tuple. La fonction devra gérer les identifiants trop grands à l'aide d'une assertion.
7. Écrire la fonction **ascendant_homme(tab : list, hommes : list) → list** qui renvoie la liste des ascendants masculins de la personne dont on a réalisé l'arbre généalogique. La fonction devra parcourir l'arbre en profondeur. De plus une simple liste Python fera office de pile.
8. Écrire une version récursive de la fonction précédente.

Exercice 4 :

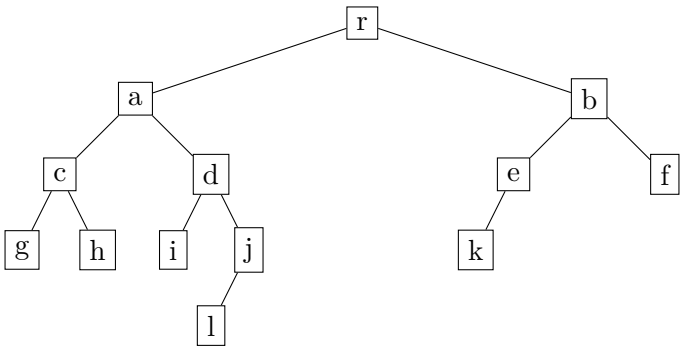

FIGURE 3 – Un arbre de chaîne de caractère

L'objectif est de créer une classe permettant de construire un arbre binaire (figure 3).

1. Créer une classe *Arbre_binaire*. On passera un paramètre *h* à son constructeur qui correspondra à la hauteur de l'arbre binaire parfait voulu. De plus, le constructeur :
   - initialisera un tableau *arbre* à la bonne taille, représentatif de l'arbre binaire. Chaque élément du tableau sera initialisé à *None*.
   - initialisera la racine avec la chaîne de caractère *"r"*.
2. Écrire la méthode **inserer(self, pere : str, fils_g : str, fils_d : str) → None** qui insère les fils gauche et droit de *pere*. La méthode lèvera une erreur d'assertion si le nœud *pere* n'a pas de fils (sort du tableau).
3. Construire l'objet *arbre_car*, instance de la classe *Arbre_binaire* puis les nœuds de la figure 3.
4. Écrire la méthode **prefixe(self, parcours : list, i : int = 0) → list :** qui renvoie la liste des nœuds traversé avec un parcours préfixe.
5. Écrire de même les méthodes *infixe* et *postfixe*.
6. *Pour les plus avancés :* transformer les méthodes de parcours précédentes pour qu'elles construisent le tableau au fur et à mesure. La signature de la méthode doit être **def prefixe(self, i : int = 0) → list :**

Christophe Viroulaud