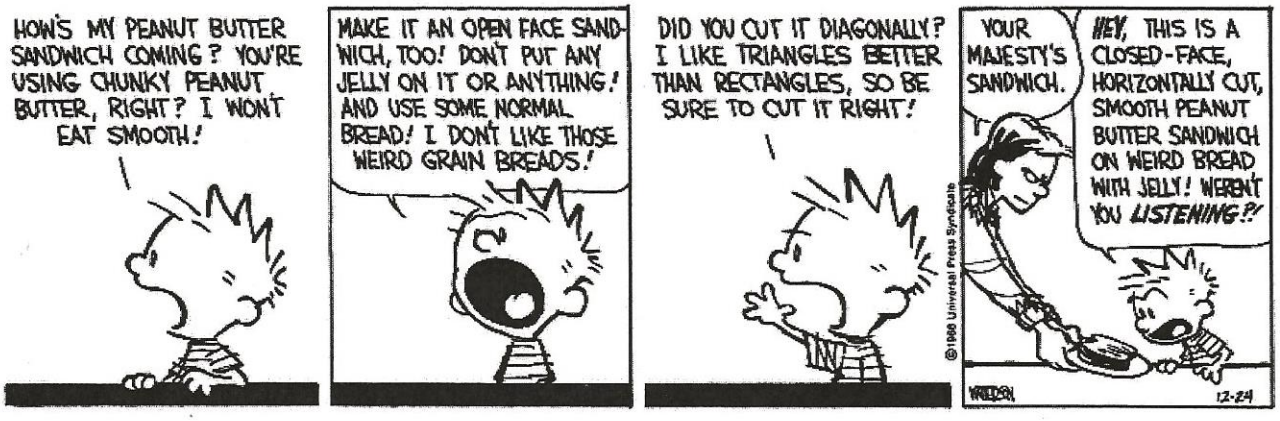
Algorithmique
Contrôle n° 2 (C2)

INFO-SUP S2
EPITA

20 février 2023 - 8 : 30

Consignes (à lire) :

- ☐ Vous devez répondre sur **les feuilles de réponses prévues à cet effet.**
  - — Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - — Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - — Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - — Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Le code :**
  - — Tout code doit être écrit dans le langage `Python` (pas de C, CAML, ALGO ou autre).
  - — **Tout code `Python` non indenté ne sera pas corrigé.**
  - — Tout ce dont vous avez besoin (fonctions, méthodes) est indiqué en **annexe** !
- ☐ Durée : 2h00



Exercice 1 (Arbre général – *2 points*)

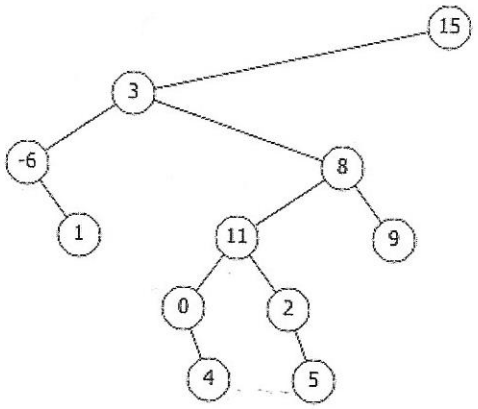


FIGURE 1 – Arbre général **T** sous forme binaire **premier fils - frère droit**

Soit l'arbre général **T** représenté sous la forme binaire **premier fils - frère droit**. Donner les clés de l'arbre général **T** ci-dessus pour chacun des ordres induits lors du parcours profondeur ci-dessous s'ils existent :

- préfixe
- infixe
- suffixe

Exercice 2 (Arbre binaire de recherche – *2 points*)

Parmi les séquences suivantes, lesquelles peuvent correspondre à la suite des étiquettes des noeuds rencontrés dans un arbre binaire de recherche lors de la recherche de 42 ?

Pour les séquences incorrectes, donner la première valeur incohérente.

① 23 - 81 - 70 - 35 - 56 - 38 - 40 - 42
② 70 - 62 - 18 - 36 - 53 - 91 - 49 - 42
③ 15 - 65 - 19 - 49 - 61 - 57 - 55 - 42
④ 28 - 32 - 33 - 81 - 55 - 45 - 37 - 42

Exercice 3 (Jeu de dames – *4 points*)

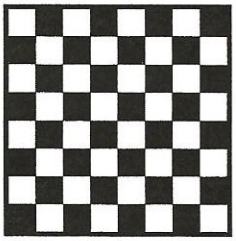


FIGURE 2 – Plateau de dames

Un plateau de dames a les caractéristiques suivantes :
- un plateau de dames a autant de lignes que de colonnes
- la case en haut à gauche est de couleur blanche
- les couleurs de ses cases alternent selon le motif de la figure 2 ci-dessus

1. Ecrire la fonction `check_line(L, n)` qui prend une liste de valeurs booléennes `L` et sa longueur `n` en paramètres, et qui vérifie si les valeurs de `L` alternent entre `True` et `False`.

```
>>> check_line([], 0)
True
>>> check_line([True], 1)
True
>>> check_line([False], 1)
True
>>> check_line([True, False, False, True], 4)
False
>>> check_line([False, True, False], 3)
True
```

2. Ecrire la fonction `check_checkers(M)` qui prend une matrice (supposée non vide) de valeurs booléennes `M` en paramètre, et qui vérifie si la matrice `M` est un plateau de dames. La valeur `True` représente la couleur blanche et la valeur `False` représente la couleur noire dans la matrice `M`.

```
>>> check_checkers([[True]])
True
>>> check_checkers([[False]])
False
>>> check_checkers([[True, False, True],[True, False, True],[True, False,
True]])
False
>>> check_checkers([[True, False, True],[False, True, False],[True, False,
True]])
True
```

Exercice 4 (Tas – *2 points*)

Ecrire la fonction `heappush(H, elt, val)` qui prend en paramètres un tas représenté par un vecteur `H`, un élément à ajouter `elt` (peut être de n'importe quel type) et sa valeur `val` (un entier), et ajoute la paire `(elt, val)` au tas `H`.

Ajout : Le nouvel élément est ajouté à la suite des autres dans le vecteur (en nouvelle feuille dans l'arbre, à la suite de la dernière feuille du dernier niveau). L'arbre reste donc parfait.

Pour rétablir la relation d'ordre, on compare la valeur du nouvel élément à celle de son père, et on les échange si nécessaire. Cette opération est répétée jusqu'à ce qu'on ait trouvé la place de l'élément (au pire comme nouvelle racine).

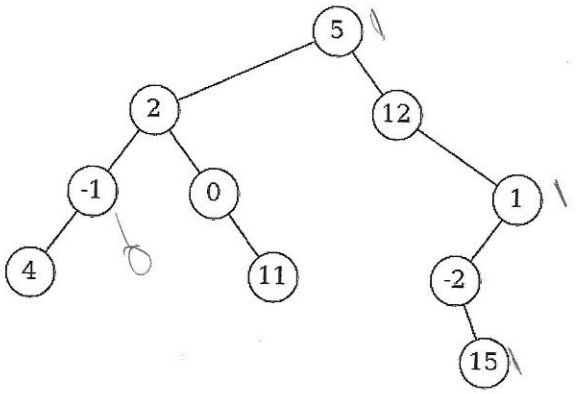


FIGURE 3 – Arbre binaire `B`

Exercice 5 (n^{ieme} noeud – *5 points*)

Ecrire la fonction `get_node(B, lvl, n)` qui prend en paramètres un arbre binaire `B`, deux entiers `lvl` et `n`, et qui retourne la clé du n^{ieme} noeud (base 0) du niveau `lvl` de `B` s'il existe, `None` sinon. Les entiers `lvl` et `n` sont supposés positifs ou nuls.

Exemples d'application sur l'arbre `B` de la figure 3 :

```
>>> get_node(None, 0, 0) #None
>>> get_node(B, 0, 0)
5
>>> get_node(B, 2, 2)
1
>>> get_node(B, 2, 5) #None
>>> get_node(B, 4, 0)
15
```

Exercice 6 (Parenté – *5 points*)

Le degré de parenté entre deux noeuds d'un arbre binaire est le nombre de liens les séparant.

Ecrire la fonction `get_kinship(B, x, y)` qui prend en paramètres un arbre binaire `B`, deux valeurs `x` et `y`, et renvoie le degré de parenté entre les noeuds de `B` contenants `x` et `y` s'ils sont sur la même branche, -1 sinon. Les clés de l'arbre binaire `B` sont supposées distinctes.

Exemples d'application sur l'arbre `B` de la figure 3 :

```
>>> get_kinship(B, 4, 11)
-1
>>> get_kinship(B, 42, 0)
-1
>>> get_kinship(B, 12, 15)
3
>>> get_kinship(B, -1, 5)
2
```

Annexes

Les arbres binaires

Les arbres binaires manipulés ici sont les mêmes qu'en td.

- L'arbre vide est `None`
- L'arbre non vide est (une référence sur) un objet de la class `BinTree` avec 3 attributs : `key`, `left`, `right`.
  - `B` : classe `BinTree`
  - `B.key` : contenu du nœud racine
  - `B.left` : le sous-arbre gauche
  - `B.right` : le sous-arbre droit

Files

Les méthodes de la classe `Queue`, que l'on suppose importée :

- `Queue()` retourne une nouvelle file ;
- q`.enqueue(`e`)` enfile e dans q ;
- q`.dequeue()` supprime et retourne le premier élément de q ;
- q`.isempty()` teste si q est vide.

Les tas

Les tas sont représentés à l'aide de la numérotation hiérarchique. Soit `H` une liste représentant un tas :

- `H[0]` n'est pas utilisé (contient None)
- `H[1]` contient la racine
- si `H[i]` est le noeud courant :
  - `H[2*i]` contient son fils gauche
  - `H[2*i+1]` contient son fils droit

Fonctions et méthodes autorisées

Vous pouvez utiliser la méthode `append` et la fonction `len` sur les listes ainsi que la fonction `range`.

Les fonctions `min` et `max`, mais uniquement avec deux valeurs entières !

Aucun opérateur n'est autorisé sur les listes (`+`, `*`, `==` ...).

Vos fonctions

Vous pouvez écrire des fonctions 'intermédiaires' / 'supplémentaires', dans ce cas vous devez donner leurs spécifications : on doit savoir ce qu'elles font.

Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.