EXO 1 :

Donner l'algorithme de parcours récursif d'un arbre en préordre :

But Construire l'algorithme itératif correspondant

a) **Visualiser sur un arbre ce qui est fait :**
b) **Que faut-il faire quand on quitte un nœud sur lequel il faudra revenir ?**
c) **Si on prend un arbre ne comportant que des fils gauches que devient l'algorithme ? Visualiser puis écrire un algo**
d) **En regardant le schéma général de cet algorithme et en voyant ce qu'il fait, trouver l'algorithme itératif correspondant**
e) **Appliquer à action préordre**
f) **En partant du résultat obtenu il ne nous reste qu'un appel récursif mais, contrairement à la première transformation, nous avons maintenant une instruction à faire derrière cet appel. Ce qui veut dire qu'il faudra récupérer la valeur de la racine de l'arbre que l'on traite, après avoir traité son sous-arbre gauche, pour aller traiter son sous-arbre droit.**
Mettre l'algorithme obtenu sous forme de schéma général

EXO 2 :

a) **Donner l'algorithme de parcours récursif d'un arbre en postordre :**
But Construire l'algorithme itératif correspondant
b) **Quelle est la différence essentielle entre les deux algos ?**
c) **En quoi cela pose-t-il un pb ?**
d) **Comment peut-on pallier ce problème ?**

question : pourquoi doit-on ajouter racine<- none ??

EXO3 : ma TODOLISTE

On veut faire une to do liste

a) **Quelle structure de données fondamentale vous parait adaptée ?**
b) **Quelques contraintes :**
    1) **Je veux pouvoir mettre une action à réaliser en priorité absolue**

2) **Je veux pouvoir quand je prends une action à réaliser la comparer avec la suivante et choisir entre les deux et remettre celle qui je ne fais pas en tête de liste, ou si vraiment je ne me sens pas une envie débordante de la réaliser en fin de liste**
3) **Vérifier la dernière tâche ajoutée**
   Quelles sont les opérations de base que je dois effectuer sur ma liste ?
   Connaissez-vous une structure de donnée adaptée ?

c) **Ecrire les fonctions de base de cette structure de données.**
   Ajout_en_tete
   Ajout_en_fin
   Retrait_en_tete
   Retrait_en_fin
d) **Ecrire les fonctions qui permettent de prendre en compte les contraintes énoncées au b) en utilisant les fonctions du c)**

EXO4 :
Suite de Fibonacci :

Suite de nombres dont chaque terme est la somme des deux précédents
Cette suite est définie récursivement par :
$U_0=0$
$U_1=1$
$U_n=U_{n-1}+U_{n-2} \quad \forall n \geq 2$
Donner l'algorithme récursif correspondant

(**)Donner la version programmation linéaire