FIT 2 [INF224] – 2021/22

Algorithmique et structures de données
TP 04

Gulfem Isiklar Alptekin – Ozgun Pinarer

Partie 1 : Arbre Binaire

Création d'un arbre de recherche binaire composé de nœuds contenant des valeurs entières et définition des opérations sur cet arbre. L'annuaire téléphonique sera mis en œuvre comme une application. Les numéros de téléphone seront placés dans l'arborescence en fonction des codes pays. Les exemples de codes de pays sont les suivants :

Almanya	49	Belçika	32	Fransa	33	Türkiye	90
Arjantin	54	Mısır	20	Hollanda	31	Isviçre	41
Avustralya	61	Monaco	377	İngiltere	44	Yunanistan	30
Avusturya	43	Brezılya	55	İspanya	34		
Meksika	52	Portekiz	351	İsveç	46		

Question 1
Définissez la structure qui décrit l'arbre de recherche binaire que vous allez créer en langage C.

Question 2
En utilisant la structure que vous avez définie, écrivez la fonction *insertNode* qui ajoute un nombre entier donné en paramètre à l'arbre de recherche à l'endroit le plus approprié dans l'arbre. Pour que;

- Si cet arbre est toujours vide, cette fonction créera un arbre à nœud unique avec sa première valeur
- S'il y a déjà des éléments dans cet arbre, cette fonction trouvera le nœud où le nouveau numéro entrant sera ajouté et l'ajoutera au bon endroit. (Utilisez l'invocation récursive pour faciliter la recherche de l'emplacement d'insertion)

Question 3
Écrivez les fonctions *findMin*, qui trouve le nœud avec le plus petit nombre dans l'arbre, et *findMax*, qui trouve le nœud avec le plus grand nombre.

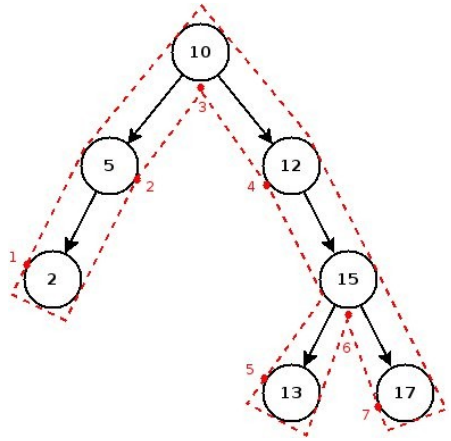
Question 4
Écrivez la fonction *deleteNode* qui supprimera un nombre entier donné en paramètre de l'arbre. Pour que;

- Cette fonction renverra une erreur si cet arbre est toujours vide
- S'il y a déjà des éléments dans cet arbre, cette fonction va d'abord trouver la place du numéro à supprimer dans l'arbre puis le

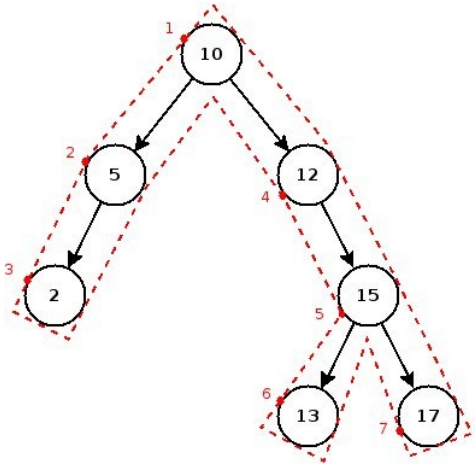
supprimer. S'il n'y a pas de numéro dans l'arbre, il renverra une erreur (utilisez un appel récursif pour faciliter la localisation du numéro. Si nécessaire, utilisez la fonction findMin que vous avez écrite à la question 3 pour le supprimer.)

Question 5

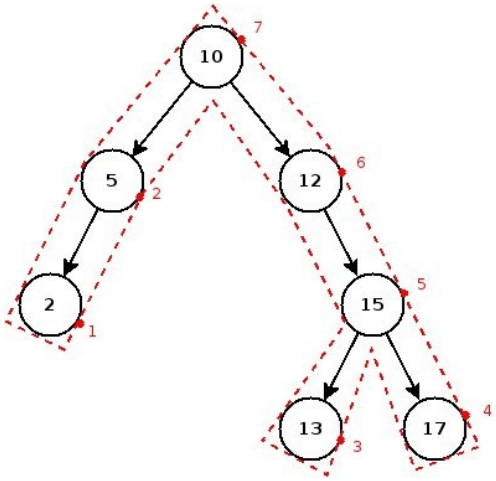
Écrivez 4 fonctions qui vont trier les nombres dans l'arbre par pre-order, in-order, post-order , level-order



Traversée in-order : 2, 5, 10, 12, 13, 15, 17



Traversée pre-order : 10, 5, 2, 12, 15, 13, 17



Traversée post-order : 2, 5, 13, 17, 15, 12, 10

Question 6
Écrivez la fonction *search* qui cherche l'info soumis par l'utilisateur.

Question 7
Pour rendre la fonction principale plus fonctionnelle, créez un menu

Partie 2 : Arbre AVL (G.M. Adelson-Velskii and E.M. Landis)

Question 8
Écrivez les fonctions suivantes :

- getHeight
- leftRotateAVL
- rightRotateAVL
- insertAVL
- deleteNodeAVL
- searchNodeAVL

Pıis créez l'arbre en ajoutant 18 pays, ensuite supprimez Meksika-Portekiz-İsvec.

IMPORTANT:

- Vous devez effectuer des études TP en tant que « fichier d'en-tête – fichier source – fichier de test » : écrivez les en-têtes des fonctions que vous avez écrites dans le fichier d'en-tête, les codes des fonctions dans le fichier source et les codes de test dans le fichier d'essai.
- Nous préférons faire des études de TP dans un environnement linux, sans avoir besoin d'IDE supplémentaire.
- Compressez vos fichiers et téléchargez-le sur le système en les nommant « NumeroEtudiant_Prenom_Nom_TPX.zip ».