ULCO **2018 - 2019**
Licence 2 **Mathématiques**

Examen – Informatique – Session 2

Documents autorisés, pas de livre, pas de calculatrice
Lundi 24 juin 2019
Durée : 2h

Exercice 1. [**3 points**] On se propose de représenter les nombres complexes à l'aide de la structure

```
struct Complexe{
   float a,b;
};
```

correspondant au nombre complexe $\mathtt{a} + \mathbf{i}\mathtt{b}$.

1. [**0.5 pt**] Ecrire une fonction `bool est_reel(Complexe z)` qui teste si le nombre complexe `z` est un réel.
2. [**1 pt**] Ecrire une fonction `conjugue` qui prend en paramètre un nombre complexe `z` et qui retourne le conjugué $\overline{\mathtt{z}}$ du nombre complexe `z`.
3. [**1.5 pts**] Ecrire une fonction `addition` qui prend en paramètres deux nombres complexes `z1` et `z2` et retourne le nombre complexe $\mathtt{z1} + \mathtt{z2}$.

Exercice 2. [**9 points**] On considère une liste chaînée de `joueur`, où `joueur` est une structure qui contient :

— un tableau `nom` de 50 caractères (`char`) ;
— un entier `nb_buts` ;
— un entier `nb_matchs` ;
— un booléen `blesse` qui vaut `true` si le joueur est blessé et `false` sinon ;
— un pointeur nommé `suivant` point sur le `joueur` suivant s'il existe et `null` sinon.

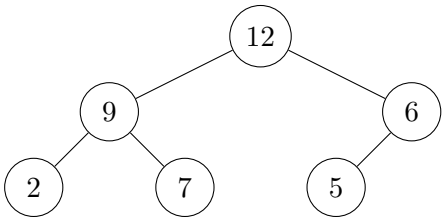
Dans cet exercice on vous demande.

1. [**1 pts**] Ecrire (en `C++`) la structure `joueur` décrite ci-dessus.
2. [**2 pts**] Ecrire une fonction `int info_blesses(joueur* liste)` retournant le nombre de joueurs blessés dans `liste`
3. [**2 pts**] Ecrire une fontion `void inserer_dernier(joueur* liste,joueur* nouveau)` qui ajoute `nouveau` en dernière position de la liste `liste`. On supposera que `nouveau` pointe sur une structure joueur déjà initialisée.
4. [**2 pts**] Ecrire une fonction

   ```
   bool tab_tries(tableau<50,char> T1,tableau<50,char> T2)
   ```

   qui retourne `true` si `T1` est avant `T2` pour l'ordre alphabétique. On supposera que si `c1` et `c2` sont deux variables de type `char` alors $\mathtt{c1} < \mathtt{c2}$ retourne `true` si le caractère `c1` est avant `c2` pour l'ordre alphabétique.
5. [**2 pts**] Ecrire une fonction `bool insere_triee(joueur* liste,joueur* nouveau)` qui, supposant que `liste` est triee pour l'ordre alphabétique des noms des joueurs, insere le joueur `nouveau` à sa place dans `liste`. On supposera que `nouveau` pointe sur une structure joueur déjà initialisée.

Exercice 3. [7 points] On note T l'arbre binaire d'entier suivant



1. **[1 pt]** Donner l'ordre de visite des sommets de T lors des parcours préfixé, infixé et postfixé.

On représente les arbres binaires d'entiers à l'aide de la structure suivante

```
struct Arbre{
  int valeur;
  Arbre* droite;
  Arbre* gauche;
};
```

2. **[1 pt]** Ecrire une fonction `int feuilles(Arbre* A)` qui retourne le nombre de feuilles de l'arbre `A`.
3. **[1 pt]** Ecrire une fonction `int hauteur(Arbre* A)` qui retourne la hauteur de l'arbre `A`.

On dit qu'un arbre binaire `A` est un tas si en tout noeud x de `A`, les valeurs des fils gauche et droite de x sont plus petites que la valeur en x. L'arbre `T` est un tas mais si on replace la valeur 2 par 10, ce n'est plus un tas.

4. **[2 pts]** Ecrire une fonction `bool est_tas(Arbre* A)` qui teste si l'arbre `A` est un tas.
5. **[0.5 pt]** Si l'arbre `A` est un tas non vide, où se trouve la plus grande valeur de `A` ? Et la seconde plus grande valeur ?
6. **[1.5 pts]** Ecrire une fonction affichant la plus petite et la plus grande valeur d'un tas non vide.