programmation examen

Nous nous proposons d'écrire la définition d'une classe *ArbreB* qui permet de stocker, dans une structure d'arbre binaire ordonné, des objets de classes dérivées de la classe abstraite *Objet*. La classe *ArbreB* possède deux fonctionnalités : l'ajout d'un élément à l'arbre, et l'affichage de l'arbre.

La classe abstraite *Objet* est définie de la façon suivante :

```
class Objet
{
      public :
           virtual int operator <(objet * o)=0;
           virtual int operator ==(objet * o)=0;
           virtual void affiche()=0;
};
```

Question 1. Ecrire la définition d'une classe *Entier*, dérivée de la classe *Objet*, et qui permet de représenter des entiers.

La classe *ArbreB* utilise des objets de la classe *Noeud* pour représenter les différents noeuds de l'arbre binaire :

```
class Noeud
{    friend class ArbreB;
     private :
          Noeud * G, *D;
          Objet * Info;
     public :
          Noeud();
          Noeud( Objet * , Noeud * g=NULL, Noeud * d=NULL);
          ~Noeud(){}
};
```

Question 2. Ecrire les définitions des constructeurs et destructeur de la classe *Noeud*.

La classe *ArbreB* est définie de la façon suivante :

```
class ArbreB
{  private :
          Noeud * racine;
          int proprietaire;
     public :
          Arbreb(int p = 1);
          ~Arbreb();
```

```
          void Ajout ( Objet * );
          void Affiche ();
};
```

Question 3. Ecrire la définition du constructeur, et du destructeur de la classe *ArbreB*. Le destructeur doit récupérer l'espace occupé par les noeuds de l'arbre, et l'espace occupé par les éléments stockés dans l'arbre si l'arbre est le propriétaire de ces éléments.

Question 4. Ecrire la définition de la fonction membre *Ajout(Objet * o)* qui ajoute l'objet *o* dans l'arbre. Pour savoir dans quel sous arbre (gauche ou droit) il faut ajouter l'élément *o*, on doit évidemment se servir des opérateurs < et == de l'objet *o*.

Question 5. Ecrire la définition de la fonction *Affiche()* qui affiche, dans l'ordre de l'arbre, les éléments stockés dans l'arbre binaire.

Soit la classe *Individu* définie de la façon suivante :

```
class Individu : public Objet
{    private :
          int numero;
          char nom[20];
     public :
          Individu(int e, char * n);
          int operator <(objet * o);
          int operator ==(objet * o);
          virtual void Affiche();
};
```

Question 6. Ecrire les définitions des constructeurs et fonctions membres de la classe *Individu*. Les opérateurs < et == portent uniquement sur le *numero* de l'individu.

Question 7. Afin de pouvoir stocker des *Individus* dans un arbre binaire ordonné, en les classant autrement que par leur numéro (par exemple par leur nom), on définit une fonction membre *Ajout(Objet *, int inf(Objet *, Objet *))* de la classe *ArbreB*. La fonction *inf* transmise en paramètre est la fonction de comparaison.

Ecrire la définition d'une fonction *int inf(Objet *o1, Objet *o2)* qui a pour valeur 1 si le nom du premier objet *o1* et plus petit (dans l'ordre lexicographique) que le nom du deuxième objet *o2*. Les deux pointeurs *o1* et *o2* sont supposés être des pointeurs sur des *Individu*.

Ecrire la définition de la fonction membre *Ajout(Objet *, int inf(Objet *, Objet *))* de la classe *ArbreB*.

Question 8. Ecrire la définition de la fonction membre de la classe *ArbreB, PourChaque(void action(Objet *))* qui exécute l'action en paramètre pour tous les éléments stockés dans l'arbre binaire.