Trier des listes

Matériel fourni :
- les notes de cours sur les listes (1) ;
- les notes de cours sur les listes (2) ;
- une unité `U_Exemples_Listes.pas`.

Objectif : Il va s'agir ici de trier des listes. Cela va être fait de deux façons

1. une fonction qui va construire une nouvelle liste triée en allouant la mémoire nécessaire pour cette construction ;
2. une procédure qui va modifier la liste à trier, sans faire aucune allocation de mémoire.

L'algorithme de tri utilisé est le tri par insertion qui s'exprime de manière récursive par les deux cas

1. trier une liste vide donne une liste vide ;
2. trier une liste de tete e et de reste l donne la liste obtenue en insérant l'élément e dans la liste obtenue en triant l.

1 Tester si une liste est triée

1.1 Préparation de l'unité `U_Element`

Dans tout ce TP, le type `ELEMENT` est égal au type `CARDINAL`.

Question 1. Dans l'unité `U_Element.pas`, ajoutez les prédicats

1. ```
   // egal(e1, e2) = VRAI si et seulement si e1 = e2
   function egal(e1,e2 : ELEMENT) : BOOLEAN;
   ```

2. ```
   // inferieur(e1, e2) = VRAI si et seulement si e1 < e2
   function inferieur(e1,e2 : ELEMENT) : BOOLEAN;
   ```

3. ```
   // inferieurOuEgal(e1, e2) = VRAI si et seulement si e1 <= e2
   function inferieurOuEgal(e1,e2 : ELEMENT) : BOOLEAN;
   ```

Question 2. Testez cette unité complétée avec un petit programme de votre choix.

1.2 Tester si une liste est triée

Question 3. Avec un algorithme récursif, réalisez le prédicat

```
// estTriee(l) = VRAI si et seulement si
// la liste l est triée
// (selon l'ordre défini par inferieurOuEgal)
function estTriee(l : LISTE) : BOOLEAN;
```

Question 4. Testez votre prédicat sur les listes $(1, 2, 3)$ et $(3, 2, 1)$.

Question 5. Testez votre prédicat sur toutes les listes définies dans l'unité `U_Exemples_Listes.pas`.

2 Insertion d'un élément dans une liste triée

Vous allez réaliser l'insertion d'un élément dans une liste.
```

2.1 La fonction insere

Question 6. Réalisez la fonction

```
// insere(e,l) = nouvelle liste avec l'élément e
// inséré au premier rang où e ne dépasse pas l'élément suivant
// si l est triée, la liste obtenue est triée
function insere(e : ELEMENT; l : LISTE): LISTE;
```

La liste passée en paramètre ne doit pas être modifiée.
Question 7. Vérifiez votre fonction sur la liste $l = (1, 3)$ en insérant les éléments 0, 2 et 4. Vérifiez que votre fonction n'a pas transformé la liste l.

2.2 La procédure inserer

Question 8. Réalisez la procédure

```
// inserer(e,l) = modifie la liste l en insérant l'élément
// e au premier rang où e ne dépasse pas l'élément suivant
// si l est triée avant, l est encore triée après
// CU : e est une liste à un seul élément
procedure inserer(const e : LISTE;var l : LISTE);
```

Cette procédure ne doit faire aucune allocation de mémoire, c'est la raison du choix du type du premier paramètre.
Question 9. Vérifiez votre procédure sur la liste $l = (1, 3)$ en insérant les éléments 0, 2 et 4.

3 Tri d'une liste

3.1 Tri avec allocation de mémoire

Question 10. Réalisez la fonction de tri

```
// trie(l) = liste triée contenant les mêmes éléments que l
function trie(l : LISTE) : LISTE;
```

Question 11. Testez votre fonction de tri avec les listes de l'unité `U_Exemples_Listes.pas`. Vérifiez en particulier que

1. les listes obtenues sont triées avec votre prédicat `estTriee` ;
2. et que les listes initiales n'ont pas été modifiées.

3.2 Tri sans allocation de mémoire

Question 12. Réalisez la procédure de tri

```
// trier(l) trie la liste l
// la liste l est modifiée
// aucune création de cellules n'est effectuée
procedure trier(var l : LISTE);
```

Question 13. Testez votre procédure de tri avec les listes de l'unité `U_Exemples_Listes.pas`. Vérifiez en particulier que les listes obtenues sont triées avec votre prédicat `estTriee`.