Algorithmique et Structures de données
(Durée 02:00)

Les réponses doivent être claires, précises et concises.

Exercice 01 : «Récursivité, Complexité & Structures de données séquentielles » (08 pts)

Soit une liste (non triée) de scores (entiers) de N joueurs, on veut connaître les K meilleurs joueurs.
Plusieurs solutions sont possibles :

- *Sol1* : On prend le meilleur (score max) on le retire de la liste et en recommence K fois.
- *Sol2* : On tri les joueurs et on affiche les K premiers.

1. **Donner** le type de cette liste.
2. **Donner** les deux solutions sous forme une procédure itérative.
3. **Donner** les deux solutions sous forme une procédure récursive.
4. **Donner** la complexité de Sol1 et Sol2 en fonction du nombre d'accès à la liste.

Avec N et K deux entiers inconnus.

Exercice 02 : « Programmation modulaire & traitement des exceptions » (07 pts)

```
Unit Uajout;
Interface
Uses UlisteEntiers,SysUtils ;
Type    EVide=class(exception) ;
        Eplein=class(exception) ;
        TclasseListe=class
        Private
                L:TListeEntiers ;
        Public
                Procedure Ajout(x: integer) ;
                Function Supp : integer ;
        end ;
Implemantation
Procedure TclasseListe.Ajout(x: integer) ;
        Begin
                ..........
        End ;
FunctionTclasseListe.Supp: integer ;
        Begin
                ..........
        End ;
End.
```

Soit *Uajout* une unité qui contient deux exceptions et une classe *TclasseListe*. Cette dernière basée sur deux méthodes :

1. *Ajout* : permet d'ajouter une valeur x si il y a une case vide dans L, sinon l'exception *Eplein* sera déclenchée.
2. *Supp* : si L n'est pas vide, le premier élément de L sera supprimé, sinon l'exception *Evide* sera déclenchée.

En utilisant *TclasseListe*, **écrire** un programme qui nous permet de :

1. introduire un ensemble d'entiers via le clavier (le nombre d'éléments est donné par l'utilisateur du programme). Il faut refaire la saisie en cas d'introduire une valeur d'un autre type.
2. Afficher cet ensemble sur écran.

Exercice 03 : « Structures de données hiérarchiques » (05 pts)

Un arbre binaire est complet si tous les niveaux excepté le dernier doivent être totalement remplis et si le dernier n'est pas totalement rempli alors il doit être rempli de gauche à droite. **Écrire** un sous-programme qui permet de vérifier si un arbre binaire R est complet ou non.