Les piles et les files
avec des structures dynamiques

Exercice 1 :
Avec l'aide de la classe « ListesDEntiers » ou directement de la classe « cellule » (testez les 2), complétez la classe « Stack » :

```
class Stack {
   …  //Ici il faut mettre des champs pour la pile
   public Stack(){} //A priori, plus besoin d'une taille limite !
   public void push(int element){}
   public void pop(){}
   public int top(){}
   public boolean isEmpty(){}    //donc, plus de isFull !
   public String toString() {}
   public int size() {}
   // Ces 2 méthodes a voir au prochain cours
   //public stack clone() {}
   //public boolean equals(Stack s) {}
}
```

Exercice 2 : Testez votre code avec un « main » où 2 piles vide doivent être créées. Vous lirez quelques entiers à mettre dans la première pile puis vous viderez cette pile en rajoutant toutes les données dans l'autre pile.

Exercice 3 : Faire les exercices proposés dans le cours ; soit en récursif, soit en itératif (boucles)

Exercice 4 : Rependre la méthode « toString » de ListeDEntier mais en récursif (pas de boucles)

Exercice 5 : Quand on manipule des piles, la méthode « size » peut être appelé très souvent. Dans le cours, nous avons vu qu'il fallait parcourir la liste pour connaître la longueur. Or, cela peut être coûteux si on recalcule cette taille a chaque fois. Une solution est de rajouter un champs « int howMany » et qui augmente ou diminue a chaque fois qu'on ajouter ou enlève un élément de la pile. Modifier votre classe Stack pour ce faire.

Exercice 6 : Écrivez les méthodes size, clone, equals pour les files comme nous l'avons fait pour les piles.