Université Bordeaux 1 Licence Informatique 2011-2012

Algorithmique 1
Contrôle continu 2

Exercice 1 *Files*

1. On implémente une file d'entiers avec une liste simplement chaînée d'entiers, avec en plus la primitive `finListe` qui permet de positionner la clé en fin de liste.
   Ecrire les primitives `valeur`, `enfiler` et `defiler` à l'aide des primitives du type abstrait des listes doublement chaînées.
2. Donner le contenu de la file après exécution de la fonction suivante.

```
fonction question_facile(): file de entier;
var F: file de entier
   debut
      creerFile(F);
      enfiler(F,1);
      enfiler(F,3);
      enfiler(F,valeur(F));
      defiler(F);
      enfiler(F,7);
      enfiler(F,3);
      defiler(F);
   fin
```

Exercice 2 *Editeur de texte*

On souhaite simuler un éditeur de texte ayant un caractère spécifique pour l'effacement de caractère (la touche backspace). On le notera par exemple #. L'éditeur dispose aussi d'une touche d'effacement de tous les caractères à gauche dans la ligne, que l'on notera par exemple $.
Appliquée à la ligne de texte : *six$06 novemm#bre 199###2007* la procédure imprimera le texte : *06 novembre 2007*.
On suppose que la ligne de texte à traiter est stockée à l'aide d'une liste simplement chainée de caractères. Ecrire la fonction Editeur, qui prend comme argument cette liste, la parcourt une seule fois, et donne en sortie une liste correpondant à la ligne de texte éditée.

Exercice 3 *Piles et listes*

1. Ecrire une fonction qui prend en entrée une pile d'entiers et qui modifie la pile en supprimant les entiers pair et fournit en sortie la liste des entiers

pairs supprimés, dans l'ordre de leut entrée dans la pile. Par exemple, la pile $[1, 4, 5, 2, 3, 6, 8[$ (où 1 est le fond de la pile) est transformée en la pile $[1, 5, 3[$ et la liste fournie en sortie est $(4, 2, 6, 8)$. Donner la complexité de l'algorithme (en justifiant brièvement).

2. Modifier l'algorithme pour que les entiers de la liste en sortie soient dans l'ordre inverse de leur entrée dans la pile (dans l'exemple ci-dessus, la sortie sera $(8, 6, 2, 4)$.

Annexe A **Liste simplement chaînée**

```
listeSC= liste de type_predefini;
```

défini en cours avec les primitives suivantes :

- **Accès**

```
fonction valeur(val L:liste d'objet) : objet;
fonction debutListe(ref L:liste d'objet) : vide;
fonction finListe(ref L:liste d'objet) : vide;
fonction suivant(ref L:liste d'objet) : vide;
fonction listeVide(val L:liste d'objet) : booleen;
fonction estFinListe(val L: liste d'objet) : booleen;
```

- **Modification**

```
fonction creerListe(ref L:liste d'objet) : vide;
fonction insererApres(ref L:liste d'objet;
      val x:objet;) : vide;
fonction insererEnTete(ref L:liste d'objet
       val x:objet) : vide;
fonction supprimerApres(ref L:liste d'objet) : vide;
fonction supprimerEnTete(ref L:liste d'objet) : vide;
fonction detruireListe(ref L:liste d'objet) : vide;
```

Annexe B **Type abstrait** *file de objet*

```
fonction valeur(val F:file de objet):objet;
fonction fileVide(val F:file de objet):booleen;
fonction creerFile(ref F:file de objet): vide;
fonction enfiler(ref F:file de objet, val v:objet):vide;
fonction defiler(ref F:file de objet):vide;
fonction detruireFile(ref F:file de objet):vide;
```

Annexe C **Implémentation du type abstrait file de objet par une liste** listeSC

```
file de objet= listeSC de objet
```

Annexe D **Type abstrait** *pile de objet*

```
fonction valeur(val P:pile de objet):objet;
fonction pileVide(val P:pile de objet):booleen;
fonction creerPile(ref P:pile de objet): vide;
fonction empiler(ref P:pile de objet, val x:objet):vide;
fonction depiler (ref P:pile de objet):vide;
fonction detruirePile(ref P:pile de objet):vide;
```