Université Bordeaux 1 Licence Informatique 2011/2012

Algorithmique 1

DS 2

Documents **non** autorisés

Exercice 2.1

Questions du cours

1. Expliquez pourquoi dans une implémentation d'une file par un tableau les indices commencent en 0 et s'arrêtent en taille-1.
2. Quelle est la différence entre une pile et une file ?

Exercice 2.2

Ecrire une fonction `PostoInfix(ref t:tableau[1..taille] de car)` qui lit une expression sous forme postfixe et retourne l'expression infixe (complétement parenthésée) qui correspond au calcul effectué par la fonction `eval-express-Postfixee` vue en TD.

N.B. L'emploi systématique des parenthèses permet de produire des expressions infixes justes. Sans les parenthèses `1 2 3 - -` et `1 2 - 3 -` conduiraient au même résultat `1 - 2 - 3` qui serait incorrect pour `1 2 3 - -`.

Exemple :

```
PreftoInfix(t[6, 3, 2, -, 1, +, /, 9, 6,-,*,'#'])  ==>
tinf[(,(,6,/,(,(,3,-,2,),+,1,),),*,(,9,-,6,),),'#']
```

Exercice 2.3

Dans l'exercice suivant, on considère le type `file d'entier`, défini en cours.

Soit u une suite finie, non vide, d'entiers, on dit que i est un maximun local pour u si

$$\forall j, 1 \leq j < i \quad \text{on a :} \quad u_j < u_i$$

A titre d'exemple la suite $12, 18, 7, 34, 7, 45, 9, 6, 4, 56, 4, 6, 2, 34$ de taille 14 possède 5 maxima locaux qui se trouvent aux rangs $1, 2, 4, 6, 10$.

1. Écrire une fonction qui à partir d'une suite stockée dans une file calcule le nombre de ses maxima locaux.
2. Quelle est sa compléxité ?

Exercice 2.4

Dans une implémentation par allocation statique du type `file d'entier` en supposant la constante `taille = 6`.

1. Donner l'état de la mémoire après la suite des instructions suivantes

   ```
   creerFile(F); enfiler(F,6);  enfiler(F,2); enfiler(F,7); enfiler(F,9);
   enfiler(F,8);defile(F); enfiler(F,5); enfiler(F,4);defiler(F);defiler(F);
   ```

2. donner le contenu de la file.

Annexe A **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;
```

Annexe B **Implémentation du type abstrait** *pile de objet* **par un tableau**

```
pile de objet= structure
                 taille: entier;
                 sommet: entier;
                 pile: tableau[1..taille] de objet
               finstructure
```

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

```
pile de objet= listeSC de objet
```

Annexe D **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 E **Implémentation du type abstrait** *file de objet* **par un tableau**

On utilise le tableau de manière circulaire avec un pointeur donnant le premie r élément et un autre donnant le dernier.

```
file de objet= structure
                 taille: entier;
                 premier: entier;
                 dernier: entier;
                 plein: booleen;
                 file: tableau [0..taille-1] de objet
               finstructure
```

Annexe F **Implémentation du type abstrait file de objet par une liste** listeDC

```
file de objet= listeDC de objet
```