Licence Informatique S3
Structures Discrètes
Aix Marseille Université 2019-2020

Devoir 3 : Données structurées

Exercice 1 Implémentation chaînée de la structure de donnée Pile
On rappelle ci-dessous la structure de donnée abstraite `Pile(T)` des piles d'éléments de type `T` :

Primitives :
- `pileVide : {} → Pile(T)`
- `estVide : Pile(T) → Booleen`
- `empiler : T × Pile(T) → Pile(T)`
- `dépiler : Pile(T) → Pile(T)`
- `sommet : Pile(T) → T`.

Préconditions :
- `dépiler(p)` *défini si* `estVide(p)` = *faux,*
- `sommet(p)` *défini si* `estVide(p)` = *faux.*

Axiomes :
- `estVide(pileVide())` = *vrai*
- `estVide(empiler(x,p))` = *faux*
- `sommet(empiler (x,p))` = `x`
- `depiler(empiler (x,p))` = `p`.

On souhaite donner une implémentation de cette SDA en utilisant les listes chaînées :

```
type MaillonT: enreg {
                  T valeur;
                  MaillonE suivant;
              }
type PileT = MaillonT;
```

Proposez une implémentation des primitives pour le type `PileT`.

Exercice 2 Suppression d'un élément dans une pile
On considère la structure de données **abstraite** `Pile(E)` rappelée dans l'exercice précédent.

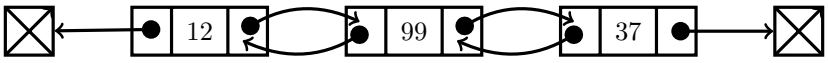
En utilisant les primitives, donnez un algorithme prenant en entrée un entier n strictement positif et une pile p et retournant la pile p dans laquelle le n-ième élément a été supprimé.

Par exemple, si $p = a_1, a_2, a_3, a_4$ et $n = 2$, l'algorithme doit retourner a_1, a_3, a_4. Si n est supérieur à la taille de la pile, la pile p est retournée inchangée.

Voici quelques indications pour vous aider : il est plus facile d'écrire cet algorithme de façon itérative (avec des boucles). Commencez par construire deux piles $rev = a_{n-1}, a_{n-2}, \cdots, a_1$ et $reste = a_{n+1}, a_{n+2}, \cdots, a_m$, puis utilisez les pour reconstruire la pile $a_1, \cdots, a_{n-1}, a_{n+1}, \cdots, a_m$.

Exercice 3 Listes doublements chaînée. Dans cet exercice, nous allons considérer des listes doublement chainées. Dans une liste chainée, on accède depuis chaque maillon au suivant dans la liste; dans une liste doublement chainée, on accède depuis chaque maillon au suivant dans la liste mais également au précédent.

Voici un exemple de liste doublement chainée d'entiers, représentant la liste $[12, 99, 37]$ (le précédent du premier maillon et le suivant du dernier maillon sont `null`):



Voici une implémentation des listes doublement chainées :

```
type MaillonEntier = enreg {
  valeur : entier;
  précédent: MaillonEntier;
  suivant: MaillonEntier;
}

type DListeEntier = MaillonEntier;
```

Une liste doublement chainée est donc soit un maillon (le premier maillon de la liste) et auquel cas elle est non vide, soit la liste vide représentée par `null`.

Ecrire un algorithme qui retire de la liste le maillon contenant le plus petit entier de la liste (s'il en existe plusieurs, c'est celui le plus proche du début de la liste qui doit être ôté).

Exercice 4

Une liste de booléens $l = b_{n-1} \dots b_0$ peut être associée de manière unique à un entier strictement positif m en utilisant le codage binaire suivant :

$$\beta(l) = 2^n + b_{n-1}2^{n-1} + \cdots + b_1 2^1 + b_0 2^0$$

Donnez une implémentation de la SDA pile de booléens en représentant chaque pile par un entier strictement positif. Cela signifie que chaque pile ne devra être codée que par un entier.