TD 1 - Types abstraits, piles et files

Objectif : Savoir dérouler un algorithme. Savoir écrire un algorithme simple - Manipuler les types abstraits

Fonctions sur les piles	**Fonctions sur les files**
last in, first out	*first in, first out*
constante pilevide $\in P$	constante filevide $\in F$
empiler : $E \times P \to P$	enfiler : $E \times F \to F$
depiler : $P \setminus \{\text{pilevide}\} \to P$	defiler : $F \setminus \{\text{filevide}\} \to F$
sommet : $P \setminus \{\text{pilevide}\} \to E$	premier : $F \setminus \{\text{filevide}\} \to E$
est_vide : $P \to \{\text{vrai}, \text{faux}\}$	est_vide : $F \to \{\text{vrai}, \text{faux}\}$

Exercice 1 - Appliquer l'algorithme suivant sur la pile $P = [5;6;7;8]$ (5 est le sommet de la pile)

Algorithme 1 : Algorithme PREMIER_DERNIER

```
1  begin
2      /* INPUT : Une pile P */
3      /* OUTPUT : La pile P modifiée */
4      if not(est_vide(P)) then
5          e := sommet(P)
6          depile(P)
7          Q := pilevide
8          while not(est_vide(P)) do
9              x := sommet(P)
10             empile(x, Q)
11             depile(P)
12         empile(e, P)
13         while not(est_vide(Q)) do
14             x := sommet(Q)
15             empile(x, P)
16             depile(Q)
17     return P
18 end
```

Exercice 2 - Appliquer l'algorithme suivant sur la file $F = [5;6;7;8]$ (8 est le premier élément de la file)

Algorithme 2 : Algorithme RENVERSEF

```
1  begin
2      /* INPUT : Une file F */
3      /* OUTPUT : La file F modifiée */
4      P := pilevide
5      while not(est_vide(F)) do
6          e := premier(F)
7          defiler(F)
8          empile(e,P)
9      while not(est_vide(P)) do
10         e := sommet(P)
11         enfile(e,F)
12         depile(P)
13     return F
14 end
```

Exercice 3 - Ecrire un algorithme permettant de *renverser* une **pile**.
RENVERSE($[5;6;7;8]$) retourne $[8;7;6;5]$.

Exercice 4 - Ecrire un algorithme qui inverse le premier et le dernier élément d'une **pile**.
INVERSEPREMDERN($[4;5;6;7;8]$) retourne $[8;5;6;7;4]$.

Exercice 5 - Ecrire un algorithme qui inverse le premier et le dernier élément d'une **file**.
INVERSEPREMDERNF($[4;5;6;7;8]$) retourne $[8;5;6;7;4]$.

Exercice 6 - On définit ci-dessous un type *liste récursive*, noté L.

- Nom du domaine : L
- Utilise les types $\mathbb{B}$ et E = type des éléments
- Signature des opérations
  - constante listeVide $\in L$
  - ajoute: $E \times L \to L$
  - retire: $L \setminus \{\text{listeVide}\} \to L$
  - tete: $P \setminus \{\text{listeVide}\} \to E$
  - est_vide: $L \to \mathbb{B}$
  - successeur: $E \times L \to E$
- Propriétés : $\forall \ell \in L, e \in E$,
  1. $\text{estVide}(\text{listeVide}) = V$
  2. $\text{tete}(\text{ajoute}(e,\ell)) = e$
  3. $\text{retire}(\text{ajoute}(e,\ell)) = \ell$
  4. $\text{succ}(\text{tete}(\ell)) = \text{tete}(\text{retire}(\ell))$
  5. $\text{ajoute}(\text{tete}(\ell),\text{retire}(\ell)) = \ell$

a) Montrer que ajouter la propriété suivante est inutile à la complétude de la définition axiomatique qui précède :

$$\text{succ}(\text{tete}(\text{ajoute}(e,\text{retire}(\ell)))) = \text{tete}(\text{retire}(\ell))$$

b) Montrer qu'ajouter la propriété suivante rendrait la définition inconsistante :

$$\text{ajoute}(\text{tete}(\ell),\ell) = \text{retire}(\ell)$$

Exercice 7 -

sorte Vecteur
utilise Entier, Elément
opérations

vect	:	Entier × Entier → Vecteur
changer-ième	:	Vecteur × Entier × Elément → Vecteur
ième	:	Vecteur × Entier → Elément
init	:	Vecteur × Entier → Booléen
borneinf	:	Vecteur → Entier
bornesup	:	Vecteur → Entier

précondition
ième$(v,\ i)$ **est-défini-ssi**
$\quad$ *borneinf*$(v) \leq i \leq$ *bornesup*(v) & *init*$(v,i) =$ vrai

axiomes
borneinf$(v) \leq i \leq$ *bornesup*$(v) \Rightarrow$
$\quad$ *ième*(*changer-ième*$(v,\ i,\ e),\ i) = e$
borneinf$(v) \leq i \leq$ *bornesup*(v) & *borneinf*$(v) \leq j \leq$ *bornesup*(v) & $i \neq j \Rightarrow$
$\quad$ *ième*(*changer-ième*$(v,\ i,\ e),\ j) =$ *ième*(v,j)

init(*vect*$(i,j),k) =$ faux
borneinf$(v) \leq i \leq$ *bornesup*$(v) \Rightarrow$
$\quad$ *init*(*changer-ième*$(v,i,e),i) =$ vrai
borneinf$(v) \leq i \leq$ *bornesup*(v) & $i \neq j \Rightarrow$
$\quad$ *init*(*changer-ième*$(v,\ i,\ e),\ j) =$ *init*(v,j)

borneinf(*vect*$(i,\ j)) = i$
borneinf(*changer-ième*$(v,\ i,\ e)) =$ *borneinf*(v)

bornesup(*vect*$(i,\ j)) = j$
bornesup(*changer-ième*$(v,\ i,\ e)) =$ *bornesup*(v)

avec
v : Vecteur; i,j,k : Entier; e : Elément

FIGURE 1 – Type abstrait Vecteur

Pour $j \neq i$, calculer ieme$\Big($changer.ieme$\big($changer.ieme$(v,i,e),j,f\big),i\Big)$. Précisez les conditions que doivent vérifier les variables.