Feuille de travaux dirigés n°5
Structures de données

Exercice 5.1 — Pile

Une **pile** est une structure de données de type LIFO (*last in first out*) : le dernier entré est le premier sorti.
On supposera que l'on dispose des primitives suivantes :

- `donnée(elt)` : renvoie la donnée associée à l'élément `elt`.
- `estVide(P)` : renvoie vrai si la pile est vide, faux sinon.
- `sommet(P)` : renvoie l'élément sommet de la pile `P`.
- `désempiler(P)` : supprime de la pile `P` le sommet.
- `empiler(P,elt)` : ajoute au sommet de la pile `P` l'élément `elt`.

Écrivez les fonctions suivantes :

1. `afficherPile(P)` : cette fonction affiche tous les éléments de la pile.
2. `désempilerKelt(P, k)` : cette fonction désempile `k` éléments si la pile contient au moins `k` éléments, sinon elle désempile toute la pile.
3. `désempilerJusquà(P,elt)` : cette fonction désempile la pile jusqu'à l'élément `elt`. L'élément `elt` n'est pas désempilé. Si l'élément n'appartient pas à la pile, alors la fonction désempile toute la pile.

Exercice 5.2 — File

Une **file** est une structure de données de type FIFO (*first in first out*) : le premier entré est le premier sorti.
On supposera que l'on dispose des primitives suivantes :

- `donnée(elt)` : renvoie la donnée associée à l'élément `elt`.
- `estVide(F)` : renvoie vrai si la file est vide, faux sinon.
- `premier(F)` : renvoie le premier élément de la file `F`.
- `défiler(F)` : supprime de la file `F` le premier élément.
- `enfiler(F, elt)` : ajoute dans la file `F` l'élément `elt`.

Écrivez les fonctions suivantes.

1. `afficherFile(F)` : cette fonction affiche tous les éléments de la file.
2. `défilerJusquà(F,elt)` : cette fonction défile la file jusqu'à l'élément `elt`. L'élément `elt` n'est pas défilé. Si l'élément n'appartient pas à la file, alors la fonction défile toute la file.

Exercice 5.3 — Pile et File

Pour cet exercice, on pourra éventuellement utiliser une ou des piles temporaires, on utilisera la primitives `créerPile()` qui renvoie une pile vide. De plus, on pourra éventuellement utiliser une ou des files temporaires, on utilisera la primitives `créerFile()` qui renvoie une file vide. Écrivez les fonctions suivantes.

1. `appartient(P,elt)` : cette fonction renvoie vrai si l'élément appartient à la pile, faux sinon. Attention, il est important que la pile ne change pas.
2. `inverser(P)` : cette fonction inverse les éléments de la pile `P`.
3. `appartient(F,elt)` : cette fonction renvoie vrai si l'élément appartient à la file, faux sinon. Attention, il est important que la file ne change pas.
4. `inverser(F)` : cette fonction inverse les éléments de la file `F`. On a le droit d'utiliser des files ou des piles temporaires.

Exercice 5.4 — Notation polonaise inverse

La texte suivant est emprunté à Wikipédia.
La Notation Polonaise Inverse (NPI) a été inventée par le philosophe australien et informaticien Charles Hamblin dans le milieu des années 1950, pour permettre les calculs sans adresse mémoire.
Avec la NPI, les opérandes précèdent l'opérateur. Cette notation permet donc de se passer des parenthèses. Par exemple, l'expression $3 \times (4 + 7)$ s'écrira $4\ 7 + 3\times$ ou bien $3\ 4\ 7 + \times$.
En pratique, sur une calculatrice de NPI, le calcul sera saisi comme suit :

"4", "entrée", "7", "+", "3", "×"

ou "3","entrée", "4", "entrée", "7", "+", "×"

La réalisation de calculatrices NPI est basée sur l'utilisation d'une pile ; c'est-à-dire, que les opérandes sont ajoutés en haut de la pile, et les résultats des calculs sont remis en haut de la pile.
Le calcul $((1 + 2) \times 4) + 3$ peut être noté en NPI comme ceci : $1\ 2 + 4 \times 3+$ ou $3\ 4\ 1\ 2 + \times +$.
L'expression est évaluée de la façon suivante (la pile est montrée après chaque opération) :

	Entrée	Opération	Pile
Étape n°1	1	Pousser l'opérande	1
Étape n°2	2	Pousser l'opérande	1, 2
Étape n°3	+	Addition	3
Étape n°4	4	Pousser l'opérande	3, 4
Étape n°5	×	Multiplication	12
Étape n°6	3	Pousser l'opérande	12, 3
Étape n°7	+	Addition	15

L'algorithme consiste donc à parcourir l'expression de gauche à droite, puis
- à empiler les opérandes rencontrés,
- à dépiler 2 opérandes à effectuer l'opération et à empiler le résultat quand un opérateur est rencontré.

Question : Écrivez de façon détaillée l'algorithme d'un calcul.

Exercice 5.5

Une maîtresse d'école décide d'instaurer un système de points correspondant à la participation des élèves. À chaque élève, est associée une pile de type LIFO qui mémorise les différentes réponses qu'il a faites. Si un élève répond bien à une question, alors un feu vert est empilé sur sa pile. Si un élève répond de façon approximative à une question, alors un feu jaune est empilé sur sa pile. Enfin, si un élève répond mal à une question, alors un feu rouge est empilé sur sa pile.
Un élève a 0 point initialement, puis les points sont comptés de la façon suivante : si la pile contient 3 fois de suite la même couleur de feu, alors le nombre de points de l'élève est incrémenté de 10 points si les feux sont verts, 5 points s'ils sont jaunes et décrémenté de 5 points s'ils sont rouges.
En outre, si 3 feux identiques apparaissent au sommet de la pile, alors les trois feux sont dépilés de la pile.

1. Détaillez le principe mis en place par la maîtresse, le nombre de points obtenus et la pile finale pour l'élève qui a reçu la suite de feux suivants (V désigne un feu vert, J un jaune et R un rouge)

   VVVJJJRRRRVVJJRRRJVVV

2. Écrivez une fonction qui prend en entrée la pile d'un élève munie des primitives vues en cours, et un tableau de feux et qui modifie la pile afin de calculer la valeur associée à l'élève et qui renvoie cette valeur. On pourra considérer que la pile est vide lors de l'appel de la fonction.