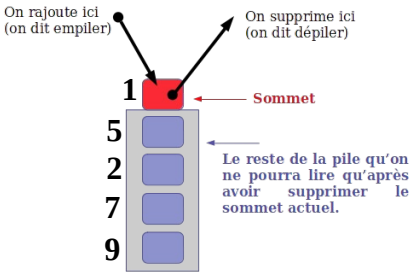
Données 15 - Type abstrait PILE – Stack - LIFO

I – Qu'est-ce que le type abstrait PILE ?

1.1 Idée générale

Une **pile** est un **type linéaire** ayant les propriétés suivantes :

- elle est composée d'une séquence _____.
- on s'impose strictement de n'agir que sur l'une des extrémités qu'on nomme le **sommet** :
  - **insérer un élément au sommet** se nomme **l'empilement**
  - **supprimer un élément au sommet** se nomme **le dépilement**



Exemple : on insère **d'abord 9**, puis 7 puis 2 puis 5, puis 1, on obtient :

LIFO :_____

1.2 Interface du type abstrait PILE

5 Fonctions primitives	Description (**cas immuable**)
`nouvellePile() -> Pile VIDE`	On crée une nouvelle pile vide `()`. En anglais, `newStack()`.
`estPileVide(p:Pile) -> bool`	Prédicat qui renvoie True si la pile **p** transmise est une pile vide. En anglais, `isEmptyStack()`
`empiler(elt:Element, p:Pile) -> Pile NON VIDE`	Renvoie une nouvelle pile où le sommet est maintenant l'élément `elt`. En anglais, ***push()***.
`depiler(p:Pile NON VIDE) -> Pile`	Renvoie une nouvelle pile après avoir supprimé le sommet. En anglais, ***pop()***.
`lireSommet(p:Pile NON VIDE) -> Element`	Renvoie le sommet de la pile **p**. Attention, on ne modifie pas la pile. En anglais, ***peek()*** *ou* ***top()***.

Exercices 01° Fournir les contenus de la pile `p1`

```
1    p1 ← nouvellePile()
2    p1 ← empiler(5, p1)
3    p1 ← empiler(7, p1)
4    x ← lireSommet(p1)
5    p1 ← empiler(x, p1)
6    p1 ← empiler(4, p1)
7    p1 ← depiler(p1)
```

Exercices 02° On veut stocker séquentiellement 1, 2, 3 dans une pile, 3 étant le sommet final.

Donner l'algorithme à utiliser et fournir la représentation finale de la pile.

Que va renvoyer la fonction `lireSommet()` ?

La pile est-elle modifiée après cette lecture ?

1.3 Type muable

Les fonctions **depiler()** et **lireSommet()** sont regroupées dans une seule fonction qui va modifier la pile en place et renvoyer le sommet supprimé.

- `empiler(elt:Elt, p:Pile) -> None` :
  on rajoute un élément au sommet sur place
- `depiler(p:Pile NON VIDE) -> Elt` :
  on supprime et renvoie la valeur au sommet

Exercices 03° Fournir les contenus successifs de la pile `p3` après exécution de chacune des instructions

```
1    p3 ← nouvellePile()
2    empiler(5, p3)
3    empiler(7, p3)
4    x ← depiler(p3)
5    empiler(x, p3)
6    empiler(4, p3)
7    depiler(p3)
```

1.4 Fonctions d'interface optionnelles

Fonctions optionnelles	Description (**cas immuable**)
`taille(p``:Pile`**`)`** `-> int :`	On renvoie le nombre d'éléments dans la Pile. Pratique pour éviter de dépasser la taile limite.
`vider(p``:Pile`**`)`** `-> Pile`	On vide la pile. Pour une pile non-mutable, on renvoie donc une pile vide. Utile pour signaler qu'on se sépare d'éléments pourtant non traités. En anglais, ***`clear()`***.
`echanger(p``:Pile TAILLE > 2`**`)`** `-> Pile TAILLE > 2`	*On inverse le sommet et l'élément juste en dessous. En anglais, ça se nomme **swap**().*

Exercices 04° Réaliser les trois algorithmes précédents pour une Pile immuable puis mutable.

II – Choix d'utilisation d'une PILE / LIFO

Exercices 05° Pour chacun des problèmes, dire si l'utilisation d'une pile est adaptée, ou pas.
DESCRIPTION PLUS COMPLETE SUR LE SITE.

A - Annulation des actions dans un logiciel.
B - Réception-gestion des requêtes HTTP
C - Réception et gestion des demandes d'impression sur une imprimante partagée
D - Inversion des éléments d'un tableau
E - Classement des produits les plus vendus
F - Gestion des clients arrivant dans un magasin
G - Stockage des fonctions lors des appels
H - Questions et réponses d'un prof lors d'un cours
I - Gestion des stocks

Exemple 1 : gestion des parenthèses ouvertes-fermantes

Exercices 06° Réaliser l'empilement et le calcul du compteur pour la séquence suivante ()))() ? Que peut-on en conclure vis à vis du parenthésage ?

Exemple 2 - Evaluation d'une expression correctement parenthésée

Exercices 07° Rechercher en utilisant cette méthode le tableau des évaluations à faire pour l'expression suivante **`(5*((5+9)+3))`**.

Exercices 08° Si on transforme la notion de parenthèses ouvrantes-fermantes en balises ouvrantes-fermantes, comment valider une page HTML dont voici les balises ?

```
1      <!DOCTYPE html>
2      <html>
3
4          <head>
5          </head>
6
7          <body>
8          </body>
9
10     </html>
```

Exercices 09° Comment invalider une page HTML dont les balises sont fournies dans cet ordre ? Vous pourrez vous demander sous quelle condition on peut rajouter une balise de fermeture.

```
1     <!DOCTYPE html>
2     <html>
3
4         <head>
5         <body>
6
7         </head>
8         </body>
9
10    </html>
```

Exemple 3 : notation polonaise inverse

Exercice 10° Ecrire la formule **`(8 + 2) * 5`** sous forme polonaise inverse.
Exercice 11° Réaliser l'interprétation progressive de cette notation.
Exercice 12° Comment écrire sous forme parenthésée normale la forme polonaise inversée **`3 8 + 4 *`** ?
Exercice 13° Réaliser l'interprétation progressive de **`3 8 + 4 *`**.

Exercice 14° En se basant sur l'empilement-dépilement des fonctions, une fois qu'on a dépilé une fonction qui répond, comment savoir à qui transmettre la réponse ?