ALGO2 – Algorithmique et Programmation 2

Fiche de TD numéro 8

Exercice 1 : Pour cet exercice, vous considérerez que la classe `Pile` et la classe `File` sont déjà écrites, et qu'elles ne contiennent que des entiers. Vous ne pouvez bien sûr utiliser que leurs méthodes. Vous pouvez utiliser d'autres variables locales de type simple ou de type `Pile` ou de type `File`. Les autres structures de données composites sont interdites (listes, tuples, dictionnaires, ...).

Spécifiez puis écrivez une fonction `retireMax(p)` qui prend en paramètre une pile `p`, lui enlève son maximum et retourne cette valeur maximale.

Le code suivant :

```
p = Pile()
for i in range(15) :
  p.empiler(randint(1,100))
print("Pile avant : ",p)
maxi = retireMax(p)
print("Pile après : ",p)
print("Maxi : ",maxi)
```

provoque le résultat suivant (attention, le sommet de la pile est le dernier élément à droite !) :

```
Pile avant :  [69, 29, 57, 34, 18, 97, 71, 34, 55, 81, 9, 30]
Pile après :  [69, 29, 57, 34, 18, 81, 71, 34, 55, 30, 9]
Maxi :  97
```

Exercice 2 : Spécifiez puis écrivez une fonction `inversePremDernFile` qui prend en paramètre une file et échange le premier et le dernier élément de la file.

Exemple : Si la file `maFile` est dans l'état suivant :

1	2	3	4	5	6

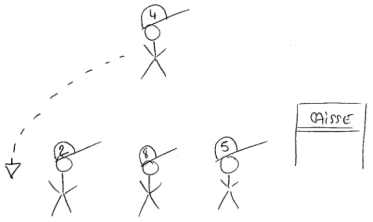
après l'appel `inversePremDern(maFile)`, `maFile` doit être dans l'état suivant :

6	2	3	4	5	1

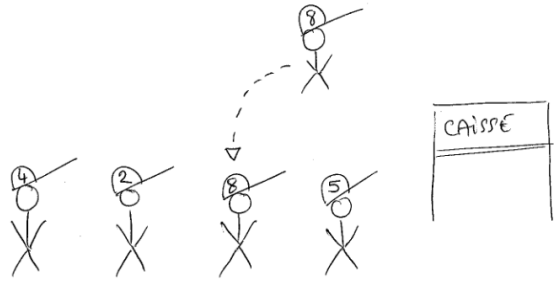
Exercice 3 : Une file un peu particulière

Au cinéma, pour aller voir un film, les gens s'ajoutent à la file d'attente devant le guichet. Mais lorsqu'une personne reconnait ses amis qui sont déjà présents dans la file, alors cette personne vient les rejoindre et s'ajoute à ce groupe. Chaque groupe passera ensemble au guichet. Nous voulons modéliser une telle file d'attente. Pour cela, nous représenterons les personnes par un numéro. Deux personnes sont amies si elles ont le même numéro.

Dans le dessin suivant, 5, 8 et 2 sont déjà dans la file d'attente (oui, tout le monde porte une casquette). 4 arrive, il devra se positionner à la fin.



Maintenant un numéro 8 arrive. Il va pouvoir se placer avec son ami 8 qui est déjà plus haut dans la file.



Q 1 . Écrivez et spécifiez une classe `FileCinema` qui représentera une file d'attente au cinéma, et qui devra être compatible avec l'exemple ci-dessous. Vous devez proposer

- un constructeur ;
- une méthode `est_vide` qui teste si la file est vide ;
- une méthode `ajoute` qui ajoute un élément (le paramètre est un numéro) dans la file : l'ajout d'un élément dans la file d'attente doit avoir le comportement décrit ci-dessus, et non le comportement habituel dans une file (l'élément n'est pas systématiquement ajouté à la fin de la liste des numéros, c'est peut-être le nombre de personnes portant ce numéro qui est incrémenté) ;
- une méthode `enleve` qui enlève un élément : elle retournera deux informations, un numéro de personne et le nombre de personnes portant ce numéro.

Exemple :

```
>>> maFile = FileCinema()
>>> maFile.ajoute(5) # ajouté au bout, numéro 5, 1 personne
>>> maFile.ajoute(8) # ajouté au bout, numéro 8, 1 personne
>>> maFile.ajoute(2) # ajouté au bout, numéro 2, 1 personne
>>> maFile.ajoute(4) # ajouté au bout, numéro 4, 1 personne
>>> maFile.ajoute(8) # le nombre de 8 va augmenter : 2 personnes
>>> maFile.enleve()
5,1
>>> maFile.enleve()
8,2
>>> maFile.enleve()
2,1
>>> maFile.enleve()
4,1
>>> maFile.estVide()
True
```

Q 2 . Écrivez et spécifiez une fonction `ouvreGuichet` qui prend en paramètre une file d'attente et le nombre de places dans la salle de cinéma. Cette fonction supprime les personnes de la file au fur et à mesure de leur traitement. Tout groupe peut entrer tant qu'il reste de la place dans la salle de cinéma. S'il ne reste pas assez de place dans la salle de cinéma pour faire entrer tout un groupe d'amis, alors tout le groupe s'en va, sans rentrer dans la salle (il y a un concert sympa pas très loin), et on passe au suivant. La fonction retourne le nombre de personnes qui sont rentrées dans la salle.