Séance 2: Calculer des calendriers

Université Paris Cité

Objectifs:

- Travailler avec des expressions arithmétiques
- Trouver des solutions efficaces pour des problèmes arithmétiques
- Nommer des calculs par des fonctions, utiliser des fonctions

1 Échauffement

Pour cette partie, rédiger les réponses aux questions dans le fichier fourni Museum.java, en modifiant le code correspondant à chaque question. Le fichier comporte des tests vous permettant de vérifier votre réponse. Vous pouvez aussi écrire vos propres tests dans le corps de la fonction `main`.

Les musées d'une ville ont tous le même fonctionnement en ce qui concerne le prix d'entrée. Les personnes qui avaient strictement moins de 20 ans l'année précédente paient cinq euros de moins que le prix normal et ceux qui ont eu 60 ans ou plus l'année précédente paient trois euros de moins que le prix normal, quant aux autres ils paient le prix normal.

Exercice 1 (Calcul de la réduction, ⋆)
Écrire une fonction `reduction` *qui prend une année de naissance et l'année courante en arguments et qui renvoie un entier valant soit 0, soit 5 soit 3 correspondant à la réduction obtenue pour la personne née dans l'année précisée en argument. Par exemple,*

```
reduction(1985, 2000) = 5
reduction(1950, 2018) = 3
reduction(1970, 2018) = 0
```

□

Exercice 2 (Calcul du prix, ⋆)
En utilisant la fonction précédente, écrire une fonction `price` *qui prend une année de naissance, l'année courante et le tarif normal du musée en arguments et qui renvoie le prix à payer. Attention le prix ne peut pas être négatif (dans ce cas il vaut 0). Par exemple,*

```
price(1985, 2000,4) = 0
price(1950, 2018,7) = 4
price(1970, 2018,16) = 16
```

□

2 Le calendrier le plus simple du monde

Pour la suite du TP, rédiger les réponses aux questions dans le fichier fourni Calendar.java, en modifiant le code correspondant à chaque question. Le fichier comporte des tests vous permettant de vérifier votre réponse. Vous pouvez aussi écrire vos propres tests dans le corps de la fonction `main`.

Le calendrier le plus simple que l'on peut imaginer consiste simplement à compter des jours à partir de 1. Nous allons appeler ce calendrier le *calendrier compteur*, le jour numéro 1 de ce calendrier est le même que le 1^{er} janvier année 1 du calendrier *grégorien*, qui est notre calendrier moderne.

Par exemple, le 14 septembre 2016 du calendrier grégorien correspond au jour 736221 du calendrier compteur (nous allons plus tard dans ce TP écrire une fonction qui permet de faire ce calcul). Ce qui montre aussi que le calendrier compteur est peu pratique pour la vie quotidienne [1].

Exercice 3 (Calculer avec le calendrier compteur, $\star$)

1. *Écrire une fonction* `diffCounter` *qui prend une date de début et une date de fin du calendrier compteur en arguments, et qui renvoie le nombre de jours entre ces deux dates, la date de début incluse et la date de fin exclue. Par exemple,*

```
diffCounter(582, 584) = 2
```

2. *On représente les jours de la semaine par des entiers : dimanche correspond à 0, lundi à 1, ..., samedi à 6. Écrire une fonction* `weekdayOfCounter` *qui prend une date compteur en argument, et qui renvoie le jour de la semaine correspondant, sachant que jour 1 était un lundi (représenté par 1). Par exemple,*

```
weekdayOfCounter(1) = 1
weekdayOfCounter(7) = 0
weekdayOfCounter(11) = 4
```

□

3 Le calendrier julien

3.1 Les années bissextiles

Dans le calendrier julien, les années bissextiles positives [2] sont les années divisible par 4. Ainsi, l'année 1789 n'est pas bissextile dans le calendrier julien, tandis que les années 1900 et 2000 le sont.

Les années bissextiles ont 366 jours et les autres 365 jours.

Remarque:
À partir de maintenant, vous devez retirer les commentaires dans la fonction `main` (en fin de fichier) pour effectuer les tests sur vos fonctions.

Exercice 4 (Les années bissextiles du calendrier julien, $\star$)

1. *Écrire une fonction* `isLeapYearJulian` *qui prend une année (≥ 0) en argument, et qui renvoie une valeur booléenne (c'est-à-dire* `true` *or* `false`*) qui indique si l'année est bissextile dans le calendrier julien. Par exemple,*

```
isLeapYearJulian(1900) = true
isLeapYearJulian(1901) = false
isLeapYearJulian(2000) = true
```

1. Pourtant, un calendrier très similaire mais avec une définition différente du jour 1 est aujourd'hui utilisé par les astronomes.
2. la règle pour les années bissextiles négatives est différente, mais ça ne doit pas nous concerner pour ce TP

2. *Écrire une fonction* `daysInYearJulian` *qui prend une année* (≥ 0) *en argument, et qui renvoie le nombre de jours dans cette année selon le calendrier julien. Indication : Servez-vous de la fonction* `isLeapYearJulian`*. Par exemple,*

```
daysInYearJulian(1900) = 366
daysInYearJulian(2000) = 366
```

□

3.2 Les mois

Si l'année est bissextile le mois de février a 29 jours, sinon il a seulement 28 jours.

Exercice 5 (Les mois dans le calendrier julien, ⋆)
Écrire une fonction `daysInMonth` *qui prend en argument un mois et un booléen qui indique s'il s'agit d'une année bissextile ou non (le booléen vaut* `true` *si on considère qu'il s'agit du mois d'une année bissextile et à* `false` *sinon), et qui renvoie le nombre de jours dans ce mois. Les mois sont numérotés à partir de 1 : janvier=1, février=2, . . ., décembre=12. Par exemple,*

```
daysInMonth(1, false)=31
daysInMonth(2, false)=28
daysInMonth(2, true)=29
daysInMonth(11, true)=30
```

□

3.3 Conversion vers le calendrier compteur

Nous allons maintenant combiner les fonctions écrites dans les exercices précédents afin de convertir des dates juliennes en des dates compteur.

Il faut d'abord savoir qu'il y a un décalage de 2 jours entre le calendrier compteur et le calendrier julien : le jour 1 du calendrier compteur est le 3 janvier de l'an 1 dans le calendrier julien.

Une solution naïve pour convertir une date julienne $(year, month, day)$ en calendrier compteur est d'additionner les durées (en nombre de jours) de toutes les années strictement plus petites que $year$, puis d'y ajouter les durées des mois strictement plus petits que $mois$, et finalement d'y ajouter day, bien sûr sans d'oublier le décalage de 2 jours.

Remarque:
À partir de maintenant, vous n'avez plus une fonction à modifier déjà donnée dans le fichier, vous devez écrire la fonction demandée en entier, en respectant le nom donné dans l'énoncé afin d'effectuer les tests correctement.

Exercice 6 (Conversion naïve de dates en compteur, ⋆⋆)
Écrire une fonction `julianToCounter` *qui prend une année, un mois et un jour en argument, et renvoie comme résultat la conversion de cette date du calendrier julien vers une date compteur. Suivez l'algorithme naïf expliqué en haut, en vous servant des fonctions écrites aux exercices précédents, et de boucles* `for`*. Par exemple,*

```
julianToCounter(1, 1, 1)=-1
julianToCounter(2, 2, 2)=396
julianToCounter(101, 1, 1)=36524
julianToCounter(2016, 9, 14)=736234
```

□

4 Le calendrier grégorien

La différence principale entre le calendrier julien et le calendrier grégorien est la définition des années *bissextiles* (en anglais : *leap year*) :

Dans le calendrier grégorien, les années divisibles par 4 sont les années bissextiles, avec une exception pour les années divisibles par 100 : Si l'année est divisible par 100 alors elle n'est *pas* bissextile, sauf si elle est divisible par 400. Ainsi, l'année 1789 n'est toujours pas bissextile dans le calendrier grégorien, l'année 1900 non plus car elle est divisible par 100 et pas divisible par 400, mais 2000 est bien une année bissextile car elle est divisible par 400.

Comme pour le calendrier julien, les années bissextiles du calendrier grégorien ont 366 jours et les autres 365 jours.

Exercice 7 (Les années bissextiles, $\star$)

1. *Écrire une fonction* `isLeapYearGregorian` *qui prend une année en argument, et qui renvoie une valeur booléenne qui indique si l'année est bissextile dans le calendrier grégorien.*

```
isLeapYearGregorian(1900) = false
isLeapYearGregorian(1901) = false
isLeapYearGregorian(2000) = true
```

2. *Écrire une fonction* `daysInYearGregorian` *qui prend une année (≥ 0) en argument, et qui renvoie le nombre de jours dans cette année selon le calendrier grégorien. Par exemple,*

```
daysInYearGregorian(1900) = 365
daysInYearGregorian(2000) = 366
```

□

4.1 Les mois

Le calendrier julien et le calendrier grégorien utilisent la même règle pour le mois de février : si l'année est bissextile le mois de février a 29 jours, sinon il a seulement 28 jours.

Exercice 8 (Les mois, $\star$)
La fonction `daysInMonth` *que vous avez rédigée dans l'exercice 5 pour le calendrier julien convient également pour le calendrier grégorien. Savez-vous expliquer pourquoi ?* □

4.2 Convertir des dates vers des dates compteurs

Heureusement, le calendrier grégorien ne souffre pas d'un décalage par rapport au calendrier compteur : le 1er janvier de l'an 1 du calendrier grégorien est le jour 1 dans le calendrier compteur.

Une solution naïve pour convertir une date grégorienne $(year, month, day)$ est d'additionner les durées (en nombre de jours) de toutes les années strictement plus petites que $year$, puis d'y ajouter les durées des mois strictement plus petits que $mois$, et finalement d'y ajouter day.

Exercice 9 (Conversion naïve de dates en compteur, $\star\star$)
Écrire une fonction `gregorianToCounter` *qui prend une année, un mois et un jour en argument, et renvoie comme résultat la conversion de cette date du calendrier grégorien vers une date compteur. Suivez l'algorithme naïf expliqué plus haut, en vous servant des fonctions écrites aux exercices précédents, et des boucles* `for`*. Par exemple,*

```
gregorianToCounter(1, 1, 1)=1
gregorianToCounter(2, 2, 2)=398
gregorianToCounter(101, 1, 1)=36525
gregorianToCounter(2016, 9, 14)=736221
```

□

Exercice 10 (Applications de la conversion, $\star$)

1. *Écrivez, en vous servant des fonctions précédentes, une fonction* `weekdayOfGregorian` *qui prend une date* $(year, month, day)$ *du calendrier grégorien en argument, et qui renvoie le jour de la semaine (représenté comme un entier entre* 0 *et* 6*, comme expliqué à l'exercice 3). Par exemple,*

```
weekdayOfGregorian(2016, 9, 14)=3
```

2. *Dans la plupart des pays européens, le passage du calendrier julien au calendrier grégorien eut lieu dans la nuit du 4 octobre 1582 (dans l'ancien calendrier julien). Il fut décrété que le jour suivant cette date serait le 15 octobre 1582 du nouveau calendrier grégorien. Vérifiez, à l'aide des fonctions que vous avez déjà écrites, que le 15 octobre 1582 grégorien était effectivement le lendemain du 4 octobre 1582 julien.*
3. *En Europe, le passage de l'heure d'hiver à l'heure d'été se fait toujours le dernier dimanche du mois de mars, selon le calendrier grégorien. Écrivez une fonction qui prend une année en entrée, et qui calcule le jour dans le mois de mars qui est le dernier dimanche de mars dans cette année. Indication : le premier jour possible est le 25. Calculez d'abord le jour de la semaine qui est le 25 mars de l'année en question, cela vous permet de trouver facilement le résultat. Par exemple,*

```
dayOfSummertime(2017)=26
```

□