# EXAMEN de TP
# session 1

*Documents autorisés : notes de cours et de TP.*
*Durée : 2 heures*

*Le barème est donné à titre indicatif et est susceptible de modifications.*

**Préambule :**
Il est conseillé (mais ce n'est pas obligatoire) de créer un dossier pour chaque partie. Comme cela, si un ajout de code fait que plus rien ne fonctionne, vous aurez toujours une version correcte des parties précédentes.
A la fin de l'examen, vous ferez une **archive** de ces dossiers que vous **déposerez sur moodle**.
Vous pouvez utiliser le programme `genere` de M Rodin (et c'est fortement conseillé).

## Partie 1 (6 points) : la classe `Date`.

**Question 1** : écrire une classe `Date` permettant de gérer une date sous la forme de 3 entiers (le jour, le mois, et l'année). Cette classe devra comporter :

- un constructeur par défaut qui initialise la date au 18 décembre 2023,
- un constructeur ayant trois entiers en paramètres,
- l'opérateur d'affichage (opérateur <<) qui affiche la date sous la forme jour/mois/année (par exemple `18/12/2023`),
- les opérateurs de copie et d'affectation (opérateur =, constructeur en fonction d'une `Date`),
- les accesseurs (`get`) et les modificateurs (`set`),
- les tests d'égalité et d'inégalité (les opérateurs == et !=)
- le destructeur.

Écrire un `main` de test, ainsi qu'un `Makefile`, et tester cette classe.

**Question 2** : écrire une (ou des) méthode(s) permettant de décaler une date de x jours (x étant un entier) avec l'opérateur +. Par exemple, si d est la date du 28/9/2023, alors d+1 sera la date du 29/9/2023 et 5+d sera la date du 3/10/2023.
Pour simplifier, on supposera que tous les mois font 30 jours, et que x est positif et inférieur à 30 (on ne demande pas de le vérifier).

## Partie 2 (4 points) : la classe `Evenement`

Une ville cherche à gérer différents événements ayant lieu chez elle, comme des concerts, des expositions, des conférences, des spectacles, etc. Pour cela, on considère une classe `Evenement` que l'on pourra ensuite dériver pour chaque type d'événement.

Écrire cette classe `Evenement`.

Elle devra contenir :

- un code de l'événement (un entier),
- un lieu (un `string`),
- les dates auxquelles il a lieu (sous la forme d'un `vector` de `Date`)
- un constructeur par défaut (qui met le code à -1),
- un constructeur avec un code et un lieu en paramètres,
- l'opérateur d'affichage (opérateur <<) qui affiche « evenement en cours de creation » si son code est -1, et le code, le lieu et les dates sinon,
- les opérateurs de copie et d'affectation (opérateur =, constructeur en fonction d'un `Evenement`),
- les tests d'égalité et d'inégalité (les opérateurs == et !=) : on testera seulement si les `Evenement` ont le même code ou non,
- le destructeur.

Écrire un `main` de test, ainsi qu'un `Makefile`, et tester cette classe.

## Partie 3 (4 points) : héritage avec la classe `Concert`

Un `Concert` est un `Evenement` ayant en plus le nom du groupe (un `string`) qui se produit. Écrire cette classe. Elle doit comporter :

- un constructeur par défaut qui met le code événement à -1,
- un constructeur ayant en paramètres un `Evenement` et un `string` (pour le nom du groupe),
- les opérateurs de copie et d'affectation (opérateur =, constructeur en fonction d'un `Concert`),
- le destructeur,
- l'opérateur << pour l'affichage, qui affiche toutes les informations concernant le `Concert`,
- les tests d'égalité et d'inégalité (les opérateurs == et !=) : on testera seulement si les `Concert` ont le même code ou non,

Écrire un `main` de test, ainsi qu'un `Makefile`, et tester cette classe.

## Partie 4 (6 points) : ajout d'une date et tarif de la publicité

**Question 1 :** écrire une (ou plusieurs) méthode(s) permettant d'ajouter une date à un `Evenement` ou à un `Concert`. La date pourra être sous la forme d'un objet de classe `Date`, ou sous la forme de 3 entiers. On ne demande pas de vérifier que la date n'a pas déjà été réservée.

Par exemple, si l'on a le morceau de code :

```
 Evenement e1(1234, « Gymnase Tom Ate»);
Date aujourdhui(18,12,2023) ;
e1.ajoutDate(aujourdhui) ;
e1.ajoutDate(25,1,2024) ;
Evenement e2(19002, « Salle Anne Anass») ;
Concert c1(e2, « Les Enerves») ;
c1.ajoutDate(10,1,2024) ;
c1.ajoutDate(20,12,2023) ;
Concert c2(Evenement(927, «Salle O. Live »),«Thierry Golo ») ;
c2.ajoutDate(aujourdhui) ;
```

```
cout << « e1 : » << e1 << endl ;
cout << « c1 : » << c1 << endl ;
cout << « c2 : » << c2 << endl ;
```

on doit obtenir (la forme de l'affichage peut différer) :

```
e1 : evnt 1234, Gymnase Tom Ate, les 18/12/2023 25/1/2024
c1 : evnt 19002, Salle Anne Anass , les 10/1/2024 20/12/2023, concert de Les Enerves
c2 : evnt 927, Salle O. Live, les 18/12 /2023, concert de Thierry Golo
```

**Question 2 :** écrire une (ou plusieurs) méthode(s) calculant le budget à prévoir pour la publicité pour un `Evenement` ou un `Concert`.
Pour un `Evenement` la publicité coûte 500€, pour un `Concert` elle coûte 200€ par jour de représentation. Écrire un `main` pour tester cette fonctionnalité.

Par exemple, si l'on complète le morceau de code de la question précédente avec

```
cout << « tarif pub pour e1 :  » << e1.pub() << endl ;
cout << « tarif pub pour c1 :  » << c1.pub() << endl ;
cout << « tarif pub pour c2 :  » << c2.pub() << endl ;
```

on obtient :

```
tarif pub pour e1 : 500
tarif pub pour c1 : 400
tarif pub pour c2 : 200
```