Université Paris 6
Licence d'informatique

Module de Programmation
ANNÉE 2000-2001

Travaux Dirigés n° 4

Un théâtre propose une formule d'adhésion à l'année à ses spectateurs, leur offrant la possibilité de réserver à l'avance leurs places pour un ensemble de spectacles.

Le but du TD consiste à gérer les inscriptions des adhérents, les places disponibles et les réservations.

Chaque spectacle est identifié par :
- Un numéro de spectacle.
- Un nom de spectacle (exemple : La flûte enchantée).
- Un prix
- Un ensemble de dates programmées, avec pour chaque date le nombre de places disponibles.

Le théâtre tient à jour une liste de ses adhérents. Chaque adhérent est identifié par :
- Nom.
- Adresse.
- Liste de réservations de l'adhérent. Chaque item de cette liste contient :
  - Numéro du spectacle réservé.
  - Date.
  - Nombre de places réservées pour ce spectacle.
- Total à payer pour toutes les réservations.

Dans tout ce qui suit, on s'interdira d'utiliser le type prédéfini `list`. Cet exercice pourra être repris plus tard en utilisant le type `list`.

1 Définir un type, nommé `typ_spec`, pour représenter un spectacle. Vous aurez à introduire d'abord un type, `typ_prog`, pour l'ensemble des dates programmées. Déclarer deux identificateurs de type `typ_spec` liés à des valeurs différentes.

2 Écrire une fonction, `reste`, qui prend en argument une valeur de type `typ_spec` et donne le total, sur l'année, des places disponibles pour ce spectacle.

3 Définir un type, `typ_theatre`, pour représenter l'ensemble des spectacles du théâtre.

4 Écrire une fonction, `prix_spec`, qui prend en premier argument une valeur de type `int` qui correspond à un numéro de spectacle, et en second argument une valeur de type `typ_theatre`. Cette fonction lève une exception avec la phrase `failwith "prix_spec"` si le numéro fourni ne figure pas dans l'ensemble des spectacles, sinon elle renvoie le prix du spectacle.

L'expression `failwith arg` est compatible avec n'importe quel type et peut donc être utilisée comme membre droit dans une définition par filtrage. De plus, son évaluation interrompt le calcul en cours et tous les calculs qui en dépendent (sauf récupération de l'exception, ce que l'on ne fera pas ici).

5 Définir un type, nommé `typ_adhérent`, pour représenter les adhérents. Vous pouvez pour cela introduire des types intermédiaires.

6 Définir un type, nommé `typ_public`, pour représenter l'ensemble des adhérents.

7 Écrire une fonction `build_adherent` qui prend en paramètres un théâtre et une liste de réservations. Cette fonction retourne le montant total des réservations. Pour cela, elle fait appel à `prix_spec`. Que se passe-t-il si un spectacle réservé n'est pas proposé par le théâtre ?

8 Écrire une fonction appelée `inscription` qui définit un nouvel adhérent. Vous pourrez pour cela définir des fonctions auxiliaires.

9 Écrire une fonction `ajouter` qui ajoute un nouvel adhérent à un ensemble d'adhérents. Elle prend en paramètres le nom de l'adhérent, son adresse, sa liste de réservations et la liste de spectacles définissant le théâtre. Pour simplifier, on supposera que le nombre de places disponibles pour un spectacle donné est toujours suffisant.

10 Écrire une fonction `rechercher` qui recherche un adhérent dont le nom est connu dans un ensemble d'adhérents.

11 Définir une fonction `total_adhesions` qui imprime le nombre courant d'adhérents et le montant perçu par le total d'inscriptions.