Colle 08

Cette huitième colle vous fera écrire une fonction simple en OCaml sur des listes, et d'autres sur un type définit manuellement. On conclut par un petit exercice en C sur des calculs trigonométriques et de l'affichage dans la console.

Ex.1 Une petite fonction sur une liste

1. Écrivez en OCaml une fonction `map` qui prend en entrée une liste `li` (de type `'a list`), une fonction `f` de type `'a -> 'b` et applique dans l'ordre `f` à chaque élément de la liste, pour produire une nouvelle liste de la même longueur. La fonction devra être récursive et ne pas utiliser `List.nth` mais un parcours récursif de la liste avec du *pattern matching*. Elle se comportera comme `List.map` ;
2. Testez la sur plusieurs listes. Que va-t-il se passer sur une liste vide ? Expliquer.

Ex.2 Un type maison pour des booléens

On cherche à écrire des fonctions en OCaml qui calculent sur des booléens, en redéfinissant le type et plein de fonctions utiles qui manipulent des booléens. On veillera à ne pas utiliser de `bool` qui sont dans la bibliothèque standard, mais uniquement les `booleen` redéfinit en question 1.

1. Définir un type énumération `booleen` qui peut valoir `True` ou `False` ;
2. Écrire une fonction `string_of_booleen` de signature `booleen -> string` qui transforme un `booleen` en chaîne de caractère ;
3. Écrire une fonction `print_booleen` de signature `booleen -> unit` qui affiche un `booleen` ;
4. Écrire une fonction `non : booleen -> booleen` unaire qui calcule la négation booléenne ;
5. Écrire une fonction `et : booleen -> booleen -> booleen` binaire qui calcule le "et" booléen ;
6. Écrire une fonction `ou : booleen -> booleen -> booleen` binaire qui calcule le "ou" booléen ;
7. Écrire une fonction `ou_exclusif : booleen -> booleen -> booleen` binaire qui calcule le "ou exclusif" booléen. Pour rappel, `x xor y` est `True` si et seulement exactement l'une des deux variables est `True` ;
8. Écrire deux fonctions `int_of_booleen : booleen -> int` et `booleen_of_int : int -> booleen`. Vous justifierez vos choix d'implémentations.

Ex.3 Des petits calculs numériques en C

Écrire une fichier C important `stdio.h` et `math.h` et qui commence par calculer une constante `pi` valant π en utilisant une formule de trigonométrie et les fonctions trigonométriques ou anti-trigonométriques. Ensuite ce fichier C devra afficher sur des lignes séparées les résultats suivants, selon le format demandé (en remplaçant `?` par sa valeur) :

 © Lilian Besson, 2021

```
pi = ?

sin(0) = ?
cos(0) = ?
tan(0) = ?

sin(pi/4) = ?
cos(pi/4) = ?
tan(pi/4) = ?

sin(pi/2) = ?
cos(pi/2) = ?
tan(pi/2) = ?

sin(3*pi/4) = ?
cos(3*pi/4) = ?
tan(3*pi/4) = ?

sin(pi) = ?
cos(pi) = ?
tan(pi) = ?
```

On rappelle qu'on compile ce fichier avec `COMPILATEUR = gcc` ou `clang` avec la ligne de commande suivante, puis on exécute le binaire produit avec la dernière ligne :

```
$ COMPILATEUR -O0 -Wall -Wextra -Wvla -Werror -fsanitize=address \
              -fsanitize=undefined -o colle8.exe colle8.c -lm
$ ./colle8.exe
```

Si une ligne affiche un résultat qui vous semble bizarre, commenter le.

 © Lilian Besson, 2021