université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

mercredi 23 octobre 2024

IPF L2 Info/LDD2 IM

TP Noté

durée 1h30 – tiers temps +30 minutes

Nom:
Prénom:
Filière/Groupe:
N° machine:

Consignes

Les seuls documents autorisés sont ceux disponibles sur la page du cours : supports de cours, aide-mémoire, énoncés et corrigés des TPs. Le barème donné est *indicatif* et pourra être modifié. Il est représentatif de la difficulté et donc du temps à passer sur chaque question. L'épreuve comporte 2 exercices.

Exercices 1 Les réponses à l'exercice 1 (QCM) sont à écrire sur l'énoncé qui devra être rendu.

Exercice 2 Récupérer sur la page du cours l'archive zip contenant le script `init_tp.sh` et le code du TP à compléter. Exécuter le code `init_tp.sh`. Ce dernier affiche votre numéro de machine que vous devrez reporter sur la liste d'émargement et sur cette feuille. Puis relancer le terminal. Compléter le fichier `tp_note_exo_2.ml`. Vous devez absolument **renseigner votre adresse email, de préférence @etu-upsaclay.fr** en tête de fichier à l'endroit prévu à cet effet. Vous pouvez à tout moment soumettre votre fichier au moyen du lien de soumission se trouvant sur la page du cours. Toutes les versions soumises seront conservées et la version la plus récente sera corrigée. **Il est de votre responsabilité de soumettre un fichier pour rendre le TP**. L'heure de dépôt est conservée, tout fichier rendu plus de 5 minutes après la notification de fin d'épreuve **ne sera pas corrigé**. Si une fonction ne compile pas, vous êtes invités **à laisser son code commenté**, et à remettre l'instruction `failwith "todo"`. Des points pourront être donnés si le code commenté est partiellement correct.

1 QCM de cours (5 points)

Pour chacune des questions, entourer la réponse correcte (vrai ou faux) :
— une bonne réponse donne 0.5 point
— une mauvaise réponse donne -0.5 point
— une absence de réponse donne 0 point

La note minimale pour une question est de 0 point (i.e. même si vous répondez faux partout, vous n'aurez pas un score négatif).

1. Le langage OCaml . . .

(a) est un langage fonctionnel	vrai	faux
(b) est un langage non typé	vrai	faux
(c) permet de faire de l'application partielle de fonction	vrai	faux

2. Donner les points communs des trois fonctions `f`, `g` et `h` ci-dessous

```
1 let f x = (x, x)
2 let g k x y = (k x, k y)
3 let rec h k x =
4     match x with
5     [] -> []
6     | u :: v -> (k u) :: k v
```

(a) les trois fonctions sont polymorphes (leur type contient des variables comme `'a`)	vrai	faux
(b) les trois fonctions sont d'ordre supérieur	vrai	faux
(c) les trois fonctions sont récursives	vrai	faux

3. On considère la fonction ci-dessous :

```
1  let rec f x y =
2    if x <= 0 then y
3    else if x mod 2 = 1 then f (x / 2) (y + 1)
4    else f (x / 2) y
```

(a) `f` est de type `int -> int`	vrai	faux
(b) `f` est récursive terminale	vrai	faux
(c) Pour `x >= 0`, `f x 0` renvoie le nombre de bits à 1 dans `x`	vrai	faux
(d) `f` est une fonction d'ordre supérieur	vrai	faux

2 Polynômes (15 points)

Le fichier `tp_note_exo2.ml` contient des tests que vous pouvez décommenter pour tester vos fonctions. Vous pouvez compiler et tester le fichier avec :

```
$ ocamlc -o tp_note_exo2.exe tp_note_exo2.ml ; ./tp_note_exo2.exe
```

On souhaite manipuler des polynômes d'une variable, par exemple

$$4X^5 - 3.5X^3 - 2$$

On rappelle que dans l'expression ci-dessus, $4X^5$, $-3.5X^2$ et 1 sont des monômes, que 4, -3.5 et 2 sont les *coefficients* de ces monômes et 5, 3 et 0 sont les *degrés* de ces monômes (en particulier $-2 = -2X^0$ est bien de degré 0). On suppose que les degrés sont positifs ou nuls.

Les polynômes sont représentés par des valeurs du types suivant :

```
1  type mono = float * int
2  type poly = mono list
```

Le type `mono` représente simplement un monôme, donné par son coefficient (un nombre flottant) et son degré. Le type `poly` représente un polynôme comme une liste de monômes.

1. (1 point) Donner une valeur OCaml correspondant au polynôme $4X^5 - 3.5X^3 - 2$.
2. (2 points) Écrire une fonction `neg_poly : poly -> poly` qui renvoie l'opposé d'un polynôme donné, c'est-à-dire le polynôme où chaque coefficient a été remplacé par son opposé. On rappelle que le « moins » unaire sur les flottants se note « `-.` ».
3. (2 points) Écrire une fonction `comp_mono : mono -> mono -> int` qui compare deux monômes. Les monômes sont d'abord comparés par degrés. S'ils sont de degré égal, alors ils sont comparés par coefficient. La fonction renvoie un nombre négatif, nul ou positif selon que son premier argument est inférieur, égal ou supérieur au second.
4. (2 points) Utiliser la fonction précédente pour écrire une fonction `sort_poly : poly -> poly` qui ordonne un polynôme selon la fonction de comparaison `comp_mono`, **en ordonant les monômes par degrés décroissants**.
5. (3 points) Écrire une fonction `reduce_poly : poly -> poly` qui prend en argument un polynôme **supposé ordonné** et qui le réduit. Les monômes de même degré sont ajoutés entre eux. Si un monôme est de coefficient 0, alors il est supprimé (ainsi, le polynôme réduit 0 est représenté par `[]`).
6. (2 points) Écrire une fonction `add_poly : poly -> poly -> poly` qui additionne deux polynômes et renvoie le résultat ordonné et réduit. On pourra dans un premier temps créer une liste de tous les monômes, puis les ordonner et les réduire.
7. (3 points) Écrire une fonction `eval_poly : poly -> float -> float` qui calcule la valeur du polynôme pour la valeur flottante de X donnée. On rappelle que l'opérateur de puissance en OCaml est « `**` ».