UJF **POLYTECH**
RICM1 **Année 2006-07**

Langages et Programmation 2, DS

Exercice 1
Votre Nom :
Votre Prénom :

1 Raccourcis clavier sous Emacs

Exercice 2 *Que font les combinaisons de touches suivantes :*
- `ESC-v` *?*
- `CTRL-x-CTRL-s` *?*
- `CTRL-x-CTRL-o` *?*

Exercice 3 *En emacs/tuareg, sans souris, comment réaliser les actions suivantes :*
- *Remonter en haut de fichier.*
- *Sauver.*
- *Interpréter la fonction courante (on suppose que l'on a déjà lancé l'interprète ocaml).*
- *Indenter le buffer entier*

Exercice 4 *Écrire une macro qui, partant de la fenêtre d'édition* `ocaml`*, va dans la fenêtre des résultats, à la fin, puis revient dans la fenêtre d'édition.*

2 Ocaml

Exercice 5 *Écrire une fonction qui prend une liste en argument, et qui renvoie, s'il existe, le* n^e *élément de la liste, et lance une exception sinon. On interdit d'utiliser* `List.nth`*.*

Exercice 6 *Écrire une fonction g à trois arguments qui retourne le maximum de ces trois arguments. En utilisant cette fonction g, écrire une fonction f à deux arguments qui retourne le maximum de 34 et de ses deux arguments. La solution demandée n'utilise pas la flèche* `->` *et elle commence par :* `let f =`

Exercice 7 *Écrire de deux façons une fonction qui prend en arguments une fonction f et une liste et qui renvoie la liste des couples $(x, f\ x)$ pour tout x élément de la liste :*

- *en utilisant le filtrage*
- *en utilisant* `List.map`

Exercice 8 *Écrire la* ***signature*** *d'un module qui contient un type abstrait t, un type ab (arbre binaire sans rien aux feuilles) et qui fournit deux fonctions :*

- *une fonction* `to_seq` *qui transforme une chaîne de caractères en le type abstrait du module,*
- *une fonction* `parse` *qui effectue l'analyse syntaxique du type abstrait.*

Comment utiliser ce module pour analyser la chaîne `"tototitii"` *?*

Exercice 9 *On fournit la grammaire G : (i désigne un entier quelconque et p est un identificateur fixé)*

$$\begin{aligned} Z &::= C \\ C &::= p\ R \\ R &::= \varepsilon \mid (P) \\ P &::= i\ Q \\ Q &::= ,\ P \mid \varepsilon \end{aligned}$$

qui reconnaît les appels de la procédure p sur un nombre quelconque de paramètres : $p(4)$, $p(1515, 42)$, *... Écrire un parseur qui compte le nombre d'arguments d'une expression respectant la grammaire G, le type lexème utilisé sera le suivant :*

```
type token =  TEnt of int | Tp | TVirgule | TParenG | TParenD
```

`Tp` *désigne le lexème associé à identificateur p, et le reste est classique.*