Consignes générales

• Indentez votre code : en suivant les suggestions de l'éditeur avec Dr Racket, ou avec F8 sur C5.
• Commentez votre code : au minimum ce que fait la fonction (en français), ainsi que **les types de ses paramètres et de son résultat**. Pensez aussi à préciser s'il y a des pré-conditions pour cette fonction (par exemple la liste ne doit pas être vide).
• Quand vous utilisez souvent une liste ou un arbre pour des tests, vous pouvez lui donner un nom pour l'utiliser en lieu et place de la liste.
• Pour le TP noté : indiquez les **tests** que vous avez effectués et leur **résultat**.

Exemple :

```
(define L '(1 2 3 4))
(define A '(5 (4 () ()) (6 () ())))

; retourne une liste dont les deux premiers éléments ont été intervertis
(define echange ;-> une liste
  (lambda (l)   ;l est une liste d'au moins 2 éléments
    (cons (cadr l) (cons (car l) (cddr l)))))

; (echange '(a b c d)) -> (b a c d)
; (echange L) -> (2 1 3 4)
```

Travail avec Dr Racket

Sauvegarde

• Sauvegardez votre fichier sur votre compte dès le début du TP.
• Faites un fichier par TP.
• Bannissez les espaces, accents et autres caractères spéciaux dans les noms de fichiers, de fonctions, de variables, etc.
• Sauvegardez régulièrement au cours du TP.

Trucs et astuces

La fenêtre d'interactions (celle du bas) se "vide" à chaque fois que l'on clique sur "exécuter". Si vous devez retaper plusieurs fois la même commande pour tester l'une de vos fonctions, cela peut vite devenir pénible. Trois solutions :

• faire du copier-coller, mais on oublie souvent de copier !
• rappeler la dernière commande avec ***Echap+P***
• écrire les appels dans la fenêtre de définitions (celle du haut). Il suffit ensuite d'exécuter pour obtenir le résultat dans la fenêtre d'interactions. Pensez ensuite à commenter les appels lorsque vous n'en avez plus besoin.

Travail avec C5 http://c5.univ-lyon1.fr

Les différentes parties du TP sont affichées à gauche, sur fond jaune. Le sujet de la question est sur la section de gauche, sur fond vert. La section du milieu sur fond blanc vous sert à écrire votre programme. La section à droite sur fond bleu permet de voir le résultat du programme.

F8 pour indenter (et exécuter le programme)

F9 pour exécuter le programme

Sauvegardez votre code en cliquant sur l'enveloppe ou Ctrl-S (plus d'explications dans la question 1 du sujet LIFAPR_TDTP01). En TP noté, il faut sauvegarder un code qui compile avant la fin du compteur horaire, car c'est le code sauvegardé qui sera noté par le correcteur.

Petites différences si vous avez l'habitude de travailler avec DrRacket :

• 3/2 est affiché comme 3/2 et pas comme 1.5 dans le résultat de l'exécution
• `false` et `true` ne sont pas reconnus, il faut mettre `#false` et `#true`
• `empty?` n'est pas reconnue, il faut utiliser `null?`
• si vous avez besoin de fonctions mathématiques, il faut ajouter `(require racket/math)` dans votre programme (mais en général c'est déjà fait)

Aide-mémoire

Formes spéciales	Syntaxe	Exemples d'utilisation

Définition d'une fonction	`(define nomDeLaFonction`
	`  (lambda (`*un ou plusieurs paramètres espacés*`)`
	`  corps de la fonction))`
		`(define carre ; -> un nombre`
		`  (lambda (a) ; a: entier`
		`  (* a a)))`

Condition : if	`(if test`
	`  ValeurSiTestVrai`
	`  ValeurSiTestFaux)`
		`(if (null? l)`
		`  0`
		`  (car l))`

Condition : cond	`(cond`
	`  (test1 valeur1)`
	`  (test2 valeur2)`
	`  ...`
	`  (else valeurN))`
		`(cond`
		`  ((< n 0) 'negatif)`
		`  ((> n 0) 'positif)`
		`  (else 'nul))`

Mémorisation : let	`(let ((identificateur1 valeur1)`
	`  (identificateur2 valeur2)`
	`  ...`
	`  (identificateurN valeurN))`
	`  expression utilisant`
	`  les identificateurs )`
		`(let ((a (sqr x))`
		`  (b (sqr y))`
		`  (c (sqr z)))`
		`  (if (< a b)`
		`  0`
		`  (+ a b c)))`

Fonctions prédéfinies	Syntaxe	Exemples d'utilisation
Opérateurs arithmétiques	`+, -, *, /`	`(+ 3 6 1)` → `10` `(- 6)` → `-6`
Opérateurs booléens	`or, and, not`	`(not #t)` → `#f` `(and #t #t #f)` → `#f`

Opérateurs de comparaison sur les nombres	`=, <, >, <=, >=`	`(= 2 4)` → `#f`
		`(< 3 9)` → `#t`
		`(>= 5 5)` → `#t`

Fonctions de comparaison	`=` (compare uniquement les nombres) `equal?` (compare deux éléments)	`(= 4 5)` → `#f` `(equal? '(5 r ) '(u 4 df 5))` → `#f`

Fonctions mathématiques	`sqr` (= carré)
	`sqrt` (= racine carrée)
	`abs` (= valeur absolue)
	`max, min`
	`modulo` (= reste de la division entière)
	`quotient` (= division entière)
		`(sqr -5)` → `25`
		`(sqrt 9)` → `3`
		`(abs -6)` → `6`
		`(max 2 6 7 5)` → `7`
		`(modulo 17 3)` → `2`
		`(quotient 341 10)` → `34`

Fonctions de test	`symbol?, number?, integer?, string?, list?, boolean?, even?` (pair), `odd?` (impair)	`(symbol? 'a)` → `#t` `(symbol? 5)` → `#f`

Fonctions sur les listes	Accès : `car, cdr`
	Construction : `cons, list, append`
	Test : `null?`
	Longueur : `length`
	Appartenance : `member?`
		`(car '(a b c)` → `a`
		`(cdr '(a b c)` → `(b c)`
		`(cons 'a '(b c)` → `(a b c)`
		`(list 'a 'b 'c)` → `(a b c)`
		`(append '(a b) '(c))` → `(a b c)`
		`(length '(a b c))` → `3`
		`(member? 'a '(a b c)` → `#t`

Fonctions diverses	`eval` (force l'évaluation) `random` (renvoie un entier aléatoire dans [0 X[)	`(eval '(+ 3 5))` → `8` `(random 5)` → `4`