Université d'Ottawa
Faculté de génie

École de science
d'informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical
Engineering
and Computer Science

Devoir 3

CSI2520 Paradigmes de programmation

Hiver 2019

À remettre le 5 avril avant 23:00

6 points

Question 1. **[1 point]**

En utilisant la fonction `map`, créer la fonction `lover` permettant de remplacer chaque nombre par son inverse (l'inverse de 0 étant 0).

```
(1over '(0 2 3 4 12 0 0 1 0))
⇨  '(0 1/2 1/3 1/4 1/12 0 0 1 0))
```

Question 2. **[2 points]**

Programmer la méthode de Newton-Rhapson permettant de trouver la racine d'une fonction à une dimension. Cette racine se trouve de façon itérative en utilisant la formule suivante :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

avec $x_n$étant l'estimé courant de la racine de *f(x)* et $f'(x_n)$ est la dérivée de la fonction évaluée à x_n.Votre fonction doit prendre trois arguments: l'estimé courant de la racine, la fonction et la dérivée de la fonction. Par exemple :

```
(newtonRhap x f fx)
```

Votre fonction doit récursivement itérer jusqu'à ce que le changement dans l'estimé de la fonction soit inférieur à une certaine tolérance définie à l'aide d'une variable globale :

```
(define TOL 1e-6)
```

Par exemple:

```
(newtonRhap 0.1 sin cos)
⇨  0
(newtonRhap 2.0 (lambda (x) (- (* x x) x 6))
```

```
                (lambda (x) (- (* 2 x) 1)))
⇨  3.0
(newtonRhap -20.0 (lambda (x) (- (* x x) x 6))
                  (lambda (x) (- (* 2 x) 1)))
⇨  -2.000000000000118
```

Question 3. **[3 points]**

Définir la fonction `p_cos` permettant de calculer le cosinus d'un angle en radians. Utiliser l'approximation suivante :

$$cos(x) = \prod_{n=1}^{\infty} \left[1 - \frac{4x^2}{pi^2(2n-1)^2}\right]$$

Le nombre de multiplications à effectuer se détermine à l'aide d'une tolérance représentant le changement minimum dans l'approximation afin de poursuivre la boucle récursive.

```
(define TOL 1e-6)
```

Par exemple:

```
(p_cos 0)
⇨  1
(p_cos (/ pi 2))
⇨  0.0
```

Utiliser une fonction auxiliaire pour vos calculs.

Question 4. **[5 points]**

Les fonctions à construire ci-dessous manipulent des listes de caractères. Vous ne devez pas utiliser ici les fonctions prédéfinies servant à traiter les chaines de caractères.

a) Définir le prédicat `separator?` retournant vrai si le caractère est un espace, une tabulation ou un changement de ligne (i.e. `#\space`, `#\tab` et `#\newline`). Le prédicat `char=?` permet la comparaison de caractères. Par exemple :

```
(separator? #\space)
⇨  #t
(separator? #\b)
⇨  #f
```

b) Écrire la fonction `cpy` permettant de copier les caractères d'une liste dans une autre liste jusqu'à ce qu'un comparateur soit rencontré.

```
(cpy '(#\H #\e #\l #\l #\o #\space #\W #\o #\r #\l #\d))
 ⇨ '(#\H #\e #\l #\l #\o)
```

c) Écrire la fonction `drop` permettant d'éliminer tous les caractères d'une liste jusqu'à ce qu'un séparateur soit rencontré.

```
(drop '(#\H #\e #\l #\l #\o #\newline #\W #\o #\r #\l #\d))
 ⇨ '(#\W #\o #\r #\l #\d)
```

d) Écrire le prédicat `same?` Comparant deux listes et retournant vrai si les caractères d'une liste sont identiques aux caractères d'une autre liste jusqu'à ce qu'un séparateur soit rencontré.

```
(same? '(#\H #\e #\l #\l #\o #\tab #\W #\o #\r #\l #\d)
       '(#\H #\e #\l #\l #\o))
 ⇨ #t

(same? '(#\H #\e #\l #\l #\o #\space #\W #\o #\r #\l #\d)
       '(#\W #\o #\r #\l #\d))
 ⇨ #f
```

e) Écrire une fonction remplaçant une liste de caractères inclue entre deux séparateurs par une autre liste de caractères.

```
(replace
'(#\a #\space #\b #\i #\r #\d #\space #\e #\a #\t #\s #\space
#\a #\space #\t #\o #\m #\a #\t #\o)
`(#\a)
`(#\t #\h #\e))
 ⇨ '(#\t #\h #\e #\space #\b #\i #\r #\d #\space #\e #\a #\t #\s
    #\space #\t #\h #\e #\space #\t #\o #\m #\a #\t #\o)
```

Ou encore:

```
(list->string (replace (string->list "a bird eats a tomato")
(string->list "a") (string->list "the")))
 ⇨ "the bird eats the tomato"
```