Université Laval
Faculté des sciences et de génie
Département d'informatique et de génie logiciel
GLO-66811

Danny Dubé
Automne 2008
Version : 17 septembre

Réponse à la série d'exercices #2
Implantation des fonctions

1.
```
(define set-intersection
  (make-clos2
   (lambda (env)
     (clos-apply2 filter
                  (make-clos1
                   (lambda (env)
                     (clos-apply2 member
                                  (list-ref env 0)
                                  (list-ref env 2)))
                   env)
                  (list-ref env 0)))
   env))
(define filter
  (make-clos2
   (lambda (env)
     (cond
       ((clos-apply1 null? (list-ref env 1))
        '())
       ((clos-apply1 (list-ref env 0)
                     (clos-apply1 car (list-ref env 1)))
        (clos-apply2 cons
                     (clos-apply1 car (list-ref env 1))
                     (clos-apply2 filter
                                  (list-ref env 0)
                                  (clos-apply1 cdr
                                               (list-ref env 1)))))
       (else
        (clos-apply2 filter
                     (list-ref env 0)
                     (clos-apply1 cdr (list-ref env 1))))))
   env))
```

2.
```
(define set-intersection
  (make-clos2-0
   (lambda (clos s1 s2)
     (clos-apply2 filter
                  (make-clos1-1
                   (lambda (clos x)
                     (clos-apply2 member x (clos-var1 clos)))
                   s2)
                  s1))))
(define filter
  (make-clos2-0
   (lambda (clos pred? lst)
     (cond
       ((clos-apply1 null? lst)
        '())
       ((clos-apply1 pred?
                     (clos-apply1 car lst))
        (clos-apply2 cons
                     (clos-apply1 car lst)
                     (clos-apply2 filter
                                  pred?
                                  (clos-apply1 cdr lst))))
       (else
        (clos-apply2 filter
                     pred?
                     (clos-apply1 cdr lst)))))))
```