$Id: 2014q4-soln1,v 1.3 2015-10-27 14:56:55-07 - - $
Answers to cmps112-2015q4-exam1, page 1

Question 1. [2]
(a) Return a value (or void).
(b) Throw an exception.
(c) Exit the program.
(d) Go into an endless loop or recursion.

Question 2(a). [2]

```
(define (filter p? list)
        (if (null? list) '()
            (let ((a (car list))
                  (fd (filter p? (cdr list))))
                 (if (p? a) (cons a fd) fd))))
```

Question 2(b). [2]

```
let rec filter p list = match list with
    | [] -> []
    | car::cdr -> if p car then car::filter p cdr
                           else filter p cdr
```

Question 3(a). [1]

```
(define (length x)
        (define (len x n)
                (if (null? x) n
                    (len (cdr x) (+ n 1))))
        (len x 0))
```

Question 3(b). [1]

```
let length x =
    let rec len x n = match x with
        | [] -> n
        | _::cdr -> len cdr (n + 1)
    in len x 0
```

Question 4. [2]

```
let rec sub' num1 num2 carry = match (num1, num2, carry) with
    | list1, [], 0 -> list1
    | list1, [], carry -> sub' list1 [- carry] 0
    | [], _::_, _ -> raise (Invalid_argument "sub'")
    | h1::t1, h2::t2, carry ->
      let diff = h1 - h2 + carry + 10
      in diff mod 10 :: sub' t1 t2 (diff / 10 - 1)
```

```
Answers to cmps112-2015q4-exam1, page 2

Question 5(a). [2]
(define (fold_left fn unit list)
        (if (null? list) unit
            (fold_left fn (fn unit (car list)) (cdr list))))

Question 5(b). [2]
let rec fold_left fn unit list = match list with
    | [] -> unit
    | car::cdr -> fold_left fn (fn unit car) cdr

Question 6(a). [2]
---- The following is 2 points, if correct.
(define (reverse list)
        (define (rev list m)
                (if (null? list) m
                    (rev (cdr list) (cons (car list) m))))
        (rev list ’()))
---- Either of the following for 3 points, if correct.
(define (snoc cdr car) (cons car cdr))
(define (reverse list) (fold_left snoc ’() list))
(define (reverse list) (fold_left (lambda (d a) (cons a d)) ’() list))

Question 6(b). [2]
---- The following is 2 points, if correct.
let reverse list =
    let rec rev’ src out = match src with
        | [] -> out
        | h::t -> rev’ t (h::out)
    in rev’ list []
---- Either of the following for 3 points, if correct.
let reverse = List.fold_left (fun tl hd -> hd::tl) []
let reverse = fold_left (fun tl hd -> hd::tl) []

Question 7. [2]
node* reverse (node* head) {
   node* out = NULL;
   while (head != NULL) {
      node* t = head;
      head = head->link;
      t->link = out;
      out = t;
   }
   return out;
}
```

```

Answers to cmps112-2015q4-exam1, page 3

 1.    (D) parametric

 2.    (C) 3

 3.    (D) int -> int -> int

 4.    (D) 4

 5.    (B) int list

 6.    (A) strong and dynamic

 7.    (B) strong and static

 8.    (D) ||

 9.    (B) (cadr '(1 2 3))

10.    (A) \lambda-calculus

11.    (C) loops

12.    (B) goto

```