COMP 348: Principles of Programming Languages
Assignment 2

Summer 2020, sections AA and AB
May 31, 2020

Achoura Bague (27877986)
Caleb Hoyne (480298)
Shadi Jiha (40131284)

Question 1

```
;gnu clisp  2.49.60

(write-line "Question 1")
(defun take-n (list n)
  (check-type n (integer 0))
  (do ((l list (cdr l))
       (r list)
       (i 0 (+ i 1)))
      ((atom l) r)
    (if (>= i n) (pop r))))

;(print ( take-n '(1 2 3) 2))
;(print ( take-n '(1 2 3) 0))
;(print ( take-n '(1 2 3 8 9 6 4 2) 4))
```

Question 2

```
(defun reverse-cut-in-half(lst)

      (setq x (REVERSE lst))              ; Reverse the list

   (list (list (car x)) (cdr x))     ; Put the head of the list in a list by itself and append it to the tail of the list

)

 (print (reverse-cut-in-half '(1 2 3)) )
```

Question 3

```
( defun flattenhelper (lst)
        (

        cond
        ((null lst) nil)
        ( (listp (car lst)) (append (flatten (car lst)) (flatten (cdr lst))))
        ( (and (null(cdr lst)) (numberp (car lst)) ) (list (car lst)))
        ( (not (numberp (car lst))) ( flatten (cdr lst)))
        ( (numberp (car lst)) ( append  (list (car lst)) (flatten (cdr lst)) ))

        )

)

( defun removealldups (list acc)
        (
        cond
        ( (member (car list) acc) (removealldups  (cdr list) acc) )
        (  (member (car list) (cdr list))  (cons (car list) (removealldups (cdr list) (cons (car
list) acc))))
        ( (null list) nil)


        ( t  (cons (car list) (removealldups (cdr list) (cons (car list) acc))))

        )

)


( defun flatten (lst)
        (
        progn
        (setf temp (flattenhelper lst))
        (setf temp (removealldups temp '()))
        temp
```

```
)
)
```

Question 4

```
( defun balancehelper(lst lngth)

        (
        cond
        ( (null lst) t)
        ( (not (listp (car lst))) (balancehelper (cdr lst) lngth ) )
        ( ( listp (car lst) )  (and (= (length (car lst)) lngth) (balancehelper (car lst) (length
(car lst) ) ) ) )




        )


)



( defun balancedp (lst)
        (
        progn
        (setf temp (length lst))
        (balancehelper lst temp)


        )
        )
```

Question 5

```
( defun bst (tree)

        (
        cond
        ( (and (null (car(cdr tree))) (null (car(cdr (cdr tree))))) t) ; both subtrees empty
        ( (and (null (car(cdr tree))) (not  (null (car(cdr (cdr tree))))) (< (car tree)
(car(car(cdr( cdr tree)))) )) (bst (car(cdr( cdr tree))))  ) ;left subtree empty
        ( (and (null (car(cdr (cdr tree)))) (not (null (car(cdr tree)))) (> (car tree) (car (car
(cdr tree)))) ) (bst (car (cdr tree))) )
        ( (and (< (car tree) (car(car(cdr( cdr tree)))) ) (> (car tree) (car (car (cdr tree)))))
(and (bst (car (cdr tree))) (bst (car(cdr( cdr tree))))) )

        )

)
```

Question 6

```
;gnu clisp  2.49.60

( defun trianglehelper (n ind)
 (if (and(typep n'integer))
   (
    cond
    ((plusp n)
        (progn
        (loop for i from 1 to n
                  do (write '*)
                  (princ " ")
                  )
        (write-line "")
        (trianglehelper (- n 1) ind)
        ))
    ((minusp n)
        (progn
           (loop for i from 1 to (+ n ind)
                do
                (princ " ")
```

```
            (princ " ")
          )

          (loop for x from n to -1
              do (write '*)
                (princ " ")
          )
          (write-line "")
          (trianglehelper (+ n 1) ind)

        )

      )


   )
    (write-line str)))

(defun triangle (n)
    (write-line "")
   (defparameter str "Invalid number; please enter a positive or negative integer")
    ; if n isn't an integer or n == 0
    (if (or (not (typep n'integer))  (= n 0) )
       ; then print error messsage
       (write-line str)
       ; else call helper
       (trianglehelper n (abs n)
   )))


;(triangle 7)
;(triangle -5)
;(triangle 2.2)
;(triangle 0)
;(triangle "hey")
```

Question 7

```
;gnu clisp  2.49.60
```

```
(defun conjucture (n)
   (cond   ( (evenp n)  (/ n 2))
           ( (oddp n) (+ (* n 3) 1))
   )
)


( defun Collatz (n)
   (CollatzHelper n '())
)


(defun CollatzHelper(n lst)

   (if (> n 1)
      (progn
         (setf n (conjucture n))
         (setf lst (append lst (list n)))
         (CollatzHelper n lst)

      )

      (loop
      (return lst)
      )
    )
)

 (print (Collatz '10))
```