Do not turn this page until you have received the signal to start.
(Please fill out the identification section above, and read the instructions below.) Good Luck!

# 1: _____/15
# 2: _____/15
# 3: _____/15

TOTAL: _____/45
Question 1.  [15 marks]

This question uses the struct:

(struct Box (value) #:mutable #:transparent)

And for diagramming purposes, treat lists as:

(struct list (first second) #:transparent)

Beside each line of code below, write down which of (A), (B), (C) and/or (D) applies, and for (A), (B) and/or (C) also include the name(s)/value(s).

(A) Creates new variable(s) named:
(B) Changes variable(s) named:
(C) Creates new Box(es) containing value(s):
(D) Changes Box(es).

Below lines with ‘define’ or ‘displayln’, draw the state of memory.
For the ones on the next page you do not need to include the parts of memory from this page.
Beside lines with ‘displayln’, show the output.

(define LoB (list (Box 148) (Box 150)))

(for ([e LoB])
  (set! e (Box (+ 1 (Box-value e)))))

(displayln LoB)
(define LoB1 (list (Box 148) (Box 150)))

(for ([e LoB1])

  (set-Box-value! e (+ 1 (Box-value e))))

(displayln LoB1)
Question 2. [15 marks]

Fill in the blanks.

A| Whether list of positive numbers `lon` adds up to number `sum`.
   An empty list is considered to add up to 0. A|
(define (sums-to? sum lon)

   (if (empty? lon) ________________

                    (sums-to? ________________

                        ________________ )))

A| A random list of positive integers adding up to nonnegative integer `sum`.
   Every possibility must eventually occur in repeated calls, but not all
   of them have to have the same probability. A|
(define (random-sums-to sum)

   (if (zero? sum) ________________

                    (let ([r ________________ ]

                        ________________ )

A| All lists of positive integers that add up to nonnegative integer `sum`.
   A|
(define (all-sums-to sum)

   (if (zero? sum) ________________

                    ________________

                    (for/list ([r ________________ ]

                        ________________

                        (for/list ([______________ ]

                            ________________ )))

Page 4 of 6

cont’d…
Question 3. [15 marks]

This question uses the struct and associated functions:

(struct Stack (contents) #:mutable)
(define (pop! s) (define top (first (Stack-contents s)))
(set-Stack-contents! s (rest (Stack-contents s)))
  top)
(define (push! s v) (set-Stack-contents! s (cons v (Stack-contents s))))

Those functions are implemented correctly.

Beside each line of code in 'swap-top!' write down which of (A), (B), (C) and/or (D) apply. Where (A), (B) and/or (C) apply also include the name/contents. Only indicate noticable effects, e.g. don’t include variables made temporarily inside the code for 'pop!'.

(A) Creates a new variable named:
(B) Changes variable named:
(C) Creates a new Stack containing contents:
(D) Changes a Stack.

(define (swap-top! s)
  (define copy s)

  (pop! s)

  (define t (pop! s))

  (push! s (pop! copy))

  (push! s t))

After the first and last lines below, draw the state of memory. Show any list(s) as just big boxes containing their numeric elements. Show the output of the ‘displayln’s.

(define s (Stack '()))

(push! s 108)
(push! s 148)
(push! s 150)
(push! s 165)
(swap-top! s)
(displayln (pop! s))
(displayln (pop! s))