# C211: Spring 2015
# Midterm: Part I (Written)

| | |
|---|---|
| Name: | ______________________ |
| Username: | ______________________ |
| Lab Section: | ______________________ |
| Lecture (circle one): | M/W     T/Th |
| Lecture Team #: | `1 2 3 4 5 6 7 8 9 10 11 12 13` |

**Instructions:** You have at most 50 minutes for the three (3) problems on this part of the exam. This part is closed-book. When you are done, give this paper to the proctor and then solve the four (4) practical problems on the computer. If you finish Part I before the 50 minute limit, you may submit your paper and move directly on to Part II without delay.

1. **Short Answer**

   (a) Draw the Box and Pointer diagram corresponding to each of the following lists.

      i. `'(a b c d)`

      ii. `'((a b) c)`

      iii. `'((a (b)) c)`

(b) Consider the following function definition:

```
(define (fascinating ls)
  (cond
    [(empty? (rest (rest ls))) (list (rest ls) (first ls))]
    [(equal? 'p (first ls)) (list (first ls) (fascinating (rest ls)))]
    [else (cons (first ls) (fascinating (rest ls)))]))
```

Write out an abbreviated **hand trace** of the application `(fascinating '(s p o c k))`. There should be one line for each recursive call until you reach the base case, and then one line for each completion step as you back out of the recursion. The first line is done for you.

```
(fascinating '(s p o c k))
== (cons 's (fascinating '(p o c k)))
```

(c) Consider the following contract and function definition:

```
; bin-op : Posn Posn -> Posn
(define (bin-op p q)
  (make-posn (* (posn-x p) (posn-x q))
             (+ (posn-y p) (posn-y q))))
```

What is the **identity** for the `bin-op` function? Justify your answer.

2. In this problem, you will design a *non-recursive* function `pig-latin` that takes a word and returns the corresponding word in *Pig Latin*, formed according to the following rule:

   If the first letter is a vowel, then append `"yay"` to the end of the word. Otherwise, move the first letter to the end of the word along with `"ay"`.

   (a) Using the following type definition, write a **contract** for the `pig-latin` function.

   ```
   A Word is a non-empty String
   ```

   (b) Write the **header** for the `pig-latin` function as it would appear at the beginning of your purpose statement. Do not write anything other than the header.

   (c) Write a **contract** and a **definition** for a predicate `vowel?` that takes a 1String and returns `true` if it corresponds to one of the five vowels in the English language (a, e, i, o, u), and `false` otherwise. Do not write anything other than the contract and definition of this predicate.

(d) Here is one `check-expect` for the `pig-latin` function. Write **two** more: one for the application `(pig-latin "most")` and one for the application `(pig-latin "illogical")`.

```
(check-expect (pig-latin "spock") "pocksay")
```

(e) Write the **definition** of the `pig-latin` function here. Don't write anything other than the definition. Make appropriate use of the `vowel?` predicate.

3. A CompoundWord is a word formed by putting two Nouns together.

```
A Noun is a Symbol
A CompoundWord is a (make-cword Noun Noun)
```

(a) Define a **structure** named `cword` for the CompoundWord type. Use `first` and `second` as the field names.

(b) Write the **contract** for each function that is created by the definition you made in part (a).

(c) Write a **data definition** for ListOfNoun and another data definition for ListOfCompoundWord.

(d) We need a function `compounder` that takes a Noun, `word`, and a list of Noun, `nouns`, and returns a list containing all compound words starting with something in `nouns` and ending with `word`, as shown in the examples below.

```
(check-expect (compounder 'house '()) '())
(check-expect (compounder 'worm '(book earth))
              (list (make-cword 'book 'worm)
                    (make-cword 'earth 'worm)))
(check-expect (compounder 'bean '(jelly lima jumping string))
              (list (make-cword 'jelly 'bean)
                    (make-cword 'lima 'bean)
                    (make-cword 'jumping 'bean)
                    (make-cword 'string 'bean)))
```

i. Write the **contract** for the `compounder` function.

ii. Write the **definition** for the `compounder` function.