Programming Assignment 1

LISP

Due: Feb 13th 11:59 p.m.

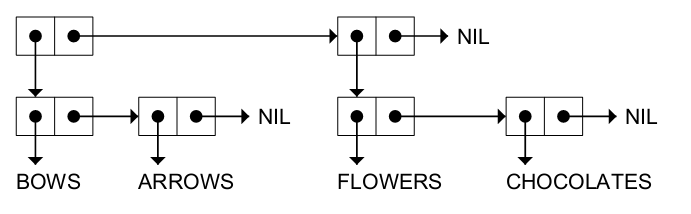
100 Points

Zip your lisp file and answers to the solutions using your NetID (cid021000.zip) and upload the file to eLearning.

Part A: Complete the questions about LISP below. Use the LISP interpreter to evaluate the commands.

1. What is the parenthesis notation for this cons cell structure?

(1)



Answer: ((BOWS ARROWS) (FLOWERS CHOCOLATES))

(2)

****

Answer: (A (B C) D)

(3)



Answer: (A B (C D))

(4)



Answer: (((PHONE HOME)))

2. Draw the internal structure for the following list(s).

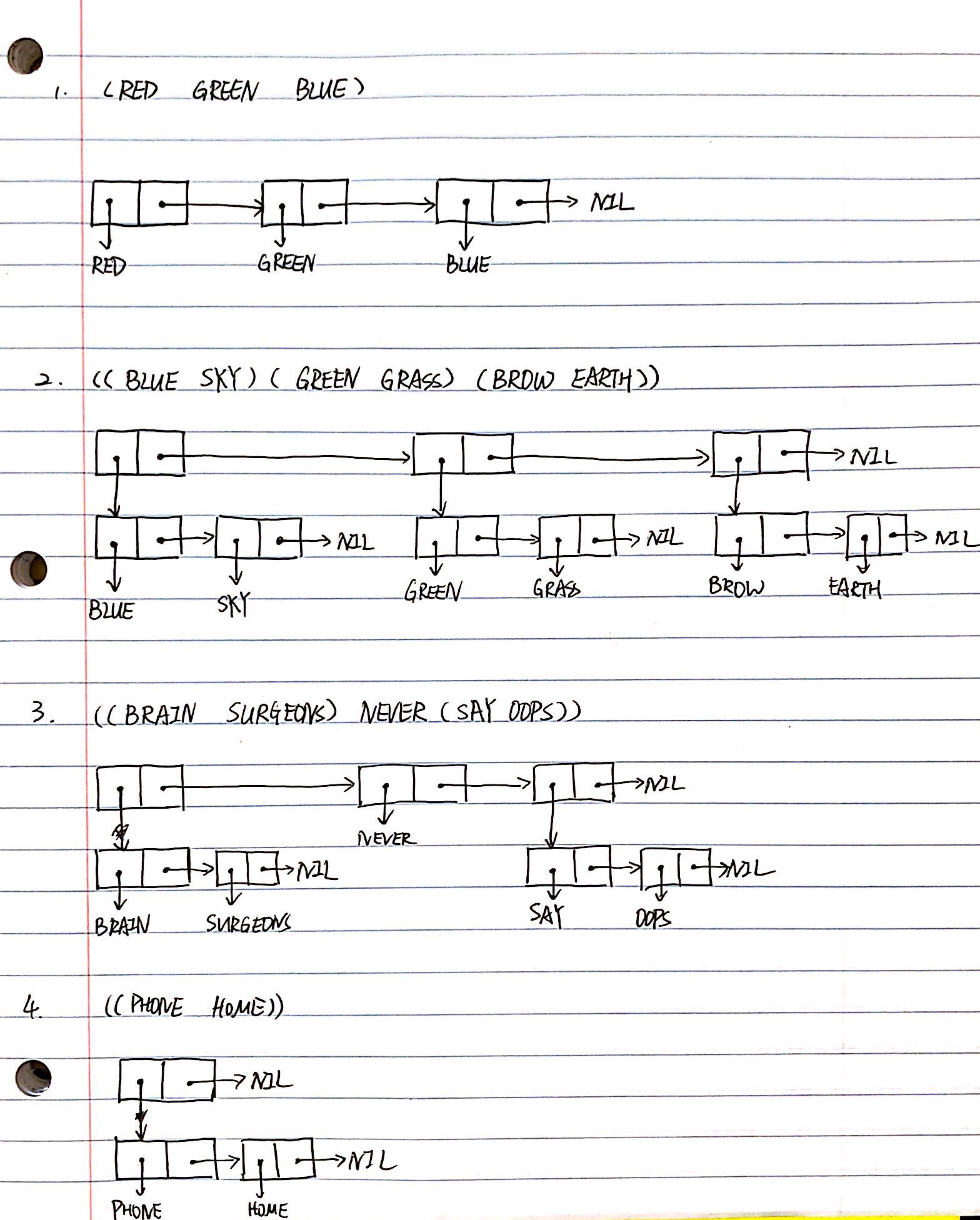
(You may draw your answer on a paper, scan or take a picture of it, and paste here)

(1) (RED GREEN BLUE)

(2) ((BLUE SKY) (GREEN GRASS) (BROWN EARTH))

(3) ((BRAIN SURGEONS) NEVER (SAY OOPS))

(4) ((PHONE HOME))



3. Consider the following list, and answer the following questions. (FIRST List) = (CAR List)

(CDR List) = (Rest List)

((BLUE CUBE) (RED PYRAMID))

|  |  |  |
| --- | --- | --- |
|  | Initial List for each problem | ((BLUE CUBE) (RED PYRAMID)) |
| 1 | CDR | ((RED PYRAMID)) |
| 2 | CADR | RED |
| 3 | CDADR | NIL |
| 4 | CADADR | NIL |
| 5 | CAR | ((BLUE CUBE)) |

4. How many elements do each of the following lists have?

|  |  |  |
| --- | --- | --- |
|  |  | Answer |
| 1 | (OPEN THE POD BAY DOORS HAL) | 6 |
| 2 | ((OPEN) (THE POD BAY DOORS) HAL) | 3 |
| 3 | ((1 2 3) (4 5 6) (7 8 9) (10 11 12)) | 4 |
| 4 | ((ONE) FOR ALL (AND (TWO (FOR ME)))) | 4 |
| 5 | ((Q SPADES)(7 HEARTS)(6 CLUBS)(5 DIAMONDS)(2 DIAMONDS)) | 5 |

5. What do each of the following expressions evaluate to?

Answer

|  |  |  |
| --- | --- | --- |
|  | Question | Answer |
| 1  2  3  4  5  6  7 | (list 'cons t nil)  (eval (list 'cons t nil))  (eval (eval (list 'cons t nil)))  (apply #'cons '(t nil))  (eval nil)  (list 'eval nil)  (eval (list 'eval nil)) | 1. (CONS T NIL) 2. (T) 3. Undefined function T 4. (T) 5. NIL 6. (EVAL NIL) 7. NIL |

6. What is the result of the following Lisp expression?

|  |  |  |
| --- | --- | --- |
|  | (setf line '(roses are red)) | Answer |
| 1 | (reverse line) | (RED ARE ROSES) |
| 2 | (first (last line)) | RED |
| 3 | (nth 1 line) | ARE |
| 4 | (reverse (reverse line)) | (ROSES ARE RED) |
| 5 | (append line (list (first line))) | (ROSES ARE RED ROSES) |
| 6 | (append (last line) line) | (RED ROSES ARE RED) |
| 7 | (list (first line) (last line)) | (ROSES (RED)) |
| 8 | (cons (last line) line) | ((RED) ROSES ARE RED) |
| 9 | (remove 'are line) | (ROSES RED) |
| 10 | (append line '(violets are blue)) | (ROSES ARE RED VIOLETS ARE BLUE) |

7. What would be the result of the following expression(s)? Answer in T or NIL.

|  |  |  |
| --- | --- | --- |
|  | (setf x1 (list 'a 'b 'c))  (setf x2 (list 'a 'b 'c))  (setf z x1) | Answer |
| 1 | (equal x1 x2) | T |
| 2 | (eq x1 x2) | NIL |
| 3 | (eq z x1) | T |
| 4 | (eq z '(a b c)) | NIL |
| 5 | (equal z '(a b c)) | T |
| 6 | (eql 'foo 'foo) | T |
| 7 | (eql 3 3.0) | NIL |
| 8 | (= 3 3.0) | T |
| 9 | (car NIL) | NIL |
| 10 | (cdr NIL) | NIL |

**Part B**

**Functional Programming**

Working individually define and test the functions described below. In doing the assignment, you may assume that the inputs to your functions have the correct data type. This means that you need not check the inputs for validity (i.e. type checking). However, your function *must* behave properly on all instances of valid input data types. You may define helper functions that are called by your primary function. In some cases, below you may be restricted to which helper functions you may use from the standard library.

**Your implementation of each function must have its function name spelled *exactly* as**

**it is in the description below (-10 points if you do not use the right function names).**

Your submission should consist of 1 source code file. Use any text editor to create .lisp file which contains your functions. Your CLISP source file should be named using your NetID. Example: cid021000.lisp. Zip your lisp file and answers to the solutions above and upload the file to eLearning.

Function 1: divisible-by-7

In CLISP define your own function that takes a single integer as an argument and returns a Boolean that indicates whether the number is divisible by 7. You do not have to perform error checking on the input.

* + **Input:** An integer
  + **Output:** A Boolean
  + **Examples:**

> (divisible-by-7 14)

#t

> (divisible-by-7 28)

#t

> (divisible-by-7 30)

#f or nil

Function 2: function-3

In CLISP define a function that takes a function as an argument and passes the number 3 to that function. The function argument must be able to accept a single integer as its argument.

* + **Input:** A named function which takes a single number as an argument.
  + **Output:** The value returned by applying the name function to the number 3.
  + **Examples:**

> (function-3 'sqrt)

1.732050807568877

> (function-3 (lambda (x) (+ x 7)))

10

> (function-3 '1+)

4

Function 3: zipper

Define a function that takes two lists as arguments and returns a single list of pairs (i.e. two element sublists). The first pair should both be the first elements from the respective lists. The second pair should be the second elements from the respective lists, and so on. If one input list is longer than the other, extra elements of the longer list are ignored. **Your implementation must be recursive.**

* **Input:** Two Lists of elements of any type, potentially heterogenous. The two lists do not have to be the same length.
* **Output:** A new list whose elements are each two-element sublists.
* **Examples:**

> (zipper '(1 2 3 4) '(a b c d))

'((1 a) (2 b) (3 c) (4 d))

> (zipper '(1 2 3) '(4 9 5 7))

'((1 4) (2 9) (3 5))

> (zipper '(5) '( ))

'( )

Function 4: my-map

In CLISP define *your own* function that duplicates the functionality of **mapcar** from the standard library. You may not use the built-in **mapcar** function as a helper function. **Your implementation must be recursive**.

* + **Input**: The input to **my-map** is a function that takes a single argument and a homogeneous list of elements of the same data type compatible with the procedure. Note: the function argument can be named or anonymous (lambda).
  + **Output**: A new list of the original elements with the same procedure applied to each.
  + **Examples**:

> (my-map 'sqrt '(9 25 81 49))

'(3 5 9 7)

> (my-map '1+ '(6 4 8 3))

'(7 5 9 4)

> (my-map (lambda (n) (\* n n)) '(5 7))

'(25 49)

> (my-map 'evenp '(2 5 7 12))

'(t NIL NIL t)

Function 5: segregate

Define a function that takes a list of integers as an argument and returns a list containing two sublists, the first sublist containing the even numbers from the original list and second sublist containing the odd numbers from the original list. **Your implementation must be recursive.**

* **Input:** A lists of Integers
* **Output:** A new list with two sublists. The first sublist contains the even numbers from the original list and second sublist contains the odd numbers.
* **Example:**

> (segregate '(7 2 3 5 8))

'((2 8) (7 3 5))

> (segregate '(3 -5 8 16 99))

'((8 16) (3 -5 99))

> (segregate '( ))

'(( ) ( ))