**CS352 QUIZ #2 (50 points) Spring 2018 Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This is an open-book test. However, you must complete it within the allowed 30 minutes time frame. You will get 5 pts penalty if submitted 10 minutes late; 10 pts off if submitted 20 minutes late – no credit afterwards. You are allowed to use Lisp and Python interpreters.

1. (10 pts) The following Lisp codes sort the list in two different ways. Please write Python codes to perform the exactly same actions. (note: you may use Python sorted() function.)

**[1]> (setf lst '((1 2) (5 6) (3 1) (2 7)))**

**[2]> (sort lst #'> :key #'first)**

**((5 6) (3 1) (2 7) (1 2))**

**[3]> (sort lst #'< :key #'second)**

**((3 1) (1 2) (5 6) (2 7))**

**#solution 1**

**lst = [[1, 2], [5, 6], [3, 1], [2, 7]]**

**newLst = sorted(lst, key=lambda x : x[0], reverse=True)**

**print(newLst)**

**#solution 2 sample student work – excellent job**

**def takeFirst (elem) :**

**return -elem[0] #note: may use reverse=True if no -**

**def takeSecond (elem) :**

**return elem[1]**

**lst\_1st\_rev = sorted (lst, key=takeFirst)**

**#note: named function can be used instead of lambda**

**lst\_2nd = sorted (lst, key=takeSecond)**

**print(lst\_1st\_rev)**

**print(lst\_2nd)**

1. (10 points) For each of the following Lisp codes, write out the result.

**(1) (member-if #’listp ’(a b (c d) e (f)))**

**Result: ((c d) e (f))**

**(2) (apply #’list ’A ’B ’C ’(D E))**

**Result: (A B C D E)**

**Note:**

**(funcall #’list ’A ’B ’C ’(D E))**

**Result: (A B C (D E))**

**(3) (apply ’(lambda (A B C) (\* A (+ B C))) ’(4 3 5))**

**Result: Error**

**(apply #’(lambda (A B C) (\* A (+ B C))) ’(4 3 5))**

**Result: 32**

**(4) (mapcar #’numberp ’(A 3 B 2 4))**

**Result: (nil t nil t t)**

**(5) (map 'list (lambda (x) (+ x 10)) '(1 2 3 4))**

**Result: error**

**(map 'list #'(lambda (x) (+ x 10)) '(1 2 3 4))**

**Result: (11 12 13 14)**

1. (10 points) Write a recursive function ColorSum in Lisp or Python that takes a list of pairs (color, value) and adds up the values of the same color. Note you need to provide an efficient implementation, i.e. walking through the list only once. For example:

**((b 3) (r 2) (g 1) (r 5) (b 2) (b 4)) -> ((b 9) (r 7) (g 1))**

**() -> ((b 0) (r 0) (g 0))**

**((r 2) (b 2) (r 5)) -> ((b 2) (r 7) (g 0))**

def colorSum (L) :

initVal = [['b', 0], ['r', 0], ['g', 0]]

print(colorSumHelper (L, initVal))

def colorSumHelper (L, iVal) :

#must have a helper function to accumulate values

sz = len(L)

if sz == 0:

return iVal

elif L[0][0] == 'b' :

iVal[0][1] += L[0][1]

elif L[0][0] == 'r' :

iVal[1][1] += L[0][1]

elif L[0][0] == 'g' :

iVal[2][1] += L[0][1]

else :

pass #wrong input just ignore

return colorSumHelper(L[1:], iVal) #recursive call

colorSum([['b', 3], ['r', 2], ['g', 1], ['r', 5], ['b', 2], ['b',4]])

colorSum([])

colorSum([['r',2],['b', 2],['r', 5]]

1. (20 pts) Write a recursiv function HowMany in Lisp or Python that takes two arguments (the first argument is a non-NIL atom while the second argument could be an atom or a list) and returns a count of the number of times the atom (i.e. first argument) occurs in the second argument at any level. For example,

**(HowMany 'A '((A (B)) A C)) -> 2**

**(HowMany 'B '((A (B)) A C)) -> 1**

**(HowMany 'D '((A (B)) A C)) -> 0**

**(HowMany 'A 'A) -> 1**

**(HowMany 'A 'B) -> 0**

(defun HowMany(A L)

(cond ((null L) 0)

((atom L)

(cond ((eq A L) 1) ;test case 4

(T 0))) ;test case 5

(T (+ (HowMany A (car L)) (HowMany A (cdr L)))) ;recursion

)

)

(HowMany 'A '((A (B)) A C))

(HowMany 'B '((A (B)) A C))

(HowMany 'D '((A (B)) A C))

(HowMany 'A 'A)

(HowMany 'A 'B)