Practice Final Solutions

You have 3 hours to complete this exam. This exam is meant solely for practice and topics that are not in this exam may be covered while topics in it may not be covered. This exam is out of 50 points.

Problem	Points
1	12
2	5
3	6
4	7
5	8
6	12
Total	50

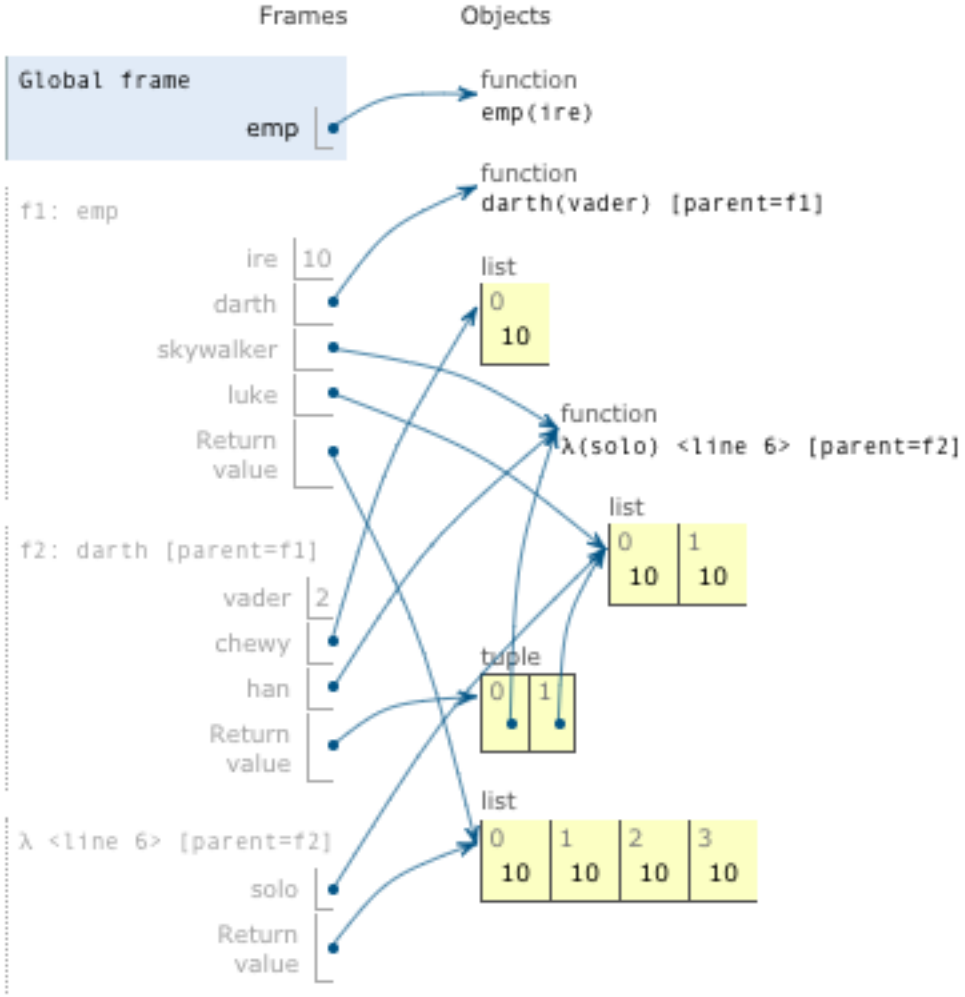
1 WWPD (12 pts)

Sorry too time constrained to do this this time around oops. Here's a picture of one of my favorite artists, Drake. Instead of finding out what Python would display please tell me what Drake is displaying- it has been eluding me for much too long.

2 Empiere strikes back (This time with Banana Cream Pie)

Fill out the environment diagram below. There at most 4 frames to create. Write the parents, variable bindings, and return values for each frame.

```
1  def emp(ire):
2      luke = ire
3      def darth(vader):
4          nonlocal luke
5          chewy = [luke]
6          han = lambda solo: solo + luke
7
8          return han, chewy*vader
9      skywalker, luke = darth(2)
10     return skywalker(luke)
11 emp(10)
```



3 Arithmetic with Python

(a) You are given an list of integers (≥ 0). Each element in the array represents the "maximum jump" from that position. For example, if 3 resides at the 1st index of your list you can move to the 1st index (no movement), 2nd index (move 1 step), 3rd index (move 2 steps), or the 4th index (move 3 steps). Your goal is to determine if you can reach the last index from the 0th index of the start of the list.

```
def canJump(nums):
        """
        >>> canJump([2,3,1,1,4])
        True
        explanation: Jump 1 step from index 0 to 1, then 3 steps to the last
        index.

        >>> canJump([3,2,1,0,4])
        False
        explanation: You will always arrive at index 3 no matter what. Its
        maximum jump length is 0, which makes it impossible to reach the
        last index.
        """
        furthest=0
        for i in range(len(nums)):
            furthest = max(furthest,i+nums[i])
            if furthest >=(len(nums)-1):
                return True
            if i==furthest:
                break
        return False
```

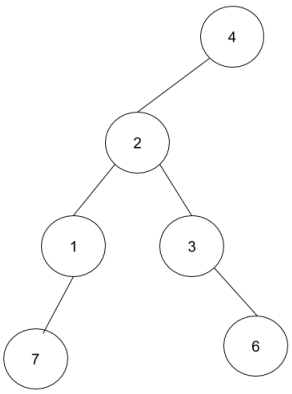
(b) You are now provided 2 LinkedLists. Your goal is to add them and return a LinkedList that represents the sum of the 2 LinkedLists. Note that the LinkedList stores the numbers in reverse order. So LinkedList $1->2->3$ represents the numbers 321.

```
class Link():
    def __init__(self,val, next = None):
        self.val = val
        self.next = next

def addTwoNumbers(l1, l2):
    """
    >>> l1 = Link(1, Link(2, Link(3)))
    >>> l2 = Link(9, Link(2))
    >>> addTwoNumbers(l1, l2)
    Link(0, Link(5,Link(3)))
    0-> 5 -> 3

    """
    def get_val(node):
        if node:
            return node.val
        return 0
    def get_next(node):
        if node:
            return node.next
        return None
    start = Link(0)
```

```
24      nde = start
25      carry = 0
26      while l1 or l1 or carry:
27          val = get_val(l1) + get_val(l2) + carry
28          carry, val = val // 10, val % 10
29          nde.next = Link(val)
30          nde = nde.next
31          l1 = get_next(l1)
32          l2 = get_next(l2)
33      return start.next
```

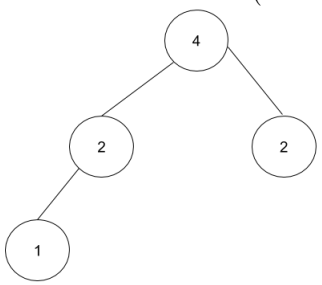
4 Small Plants

(a) Given a Binary Tree, you want to compute the diameter of the Tree. We define the diameter as the longest length between any 2 nodes in the Tree. As a note, it is not necessary for the diameter to pass through the root of the tree. For example, the diameter of the below tree is 5. The longest path would start tat the bottom left left (7) move up 1 up to 2 down to 3 and then down to 6.




```
1  class Tree():
2      def __init__(self, val, left = None, right = None):
3          self.val = val
4          self.left, self.right = left, right
5
6  def diameterOfBinaryTree(root):
7      max_diam = 0
8      def treeDepth(root):
9          nonlocal max_diam
10         if root is None:
11             return 0
12         left_depth = treeDepth(root.left)
13         right_depth = treeDepth(root.right)
14         #Making the maximum diameter so far the depth_left+depth_right+1
15         #or the max_length so far
16         max_diam = max(max_diam, left_depth + right_depth + 1)
17         #returning the height of the Tree.
18         return max(left_depth, right_depth) + 1
19     treeDepth(root)
20     return max_diam
```

(b) We know want to find the width of a Binary Search Tree. The width is defined as the number of nodes on the level with the most nodes. For example, the width of the below Tree is 2 because the maximum amount of nodes on a level is 2 (the 2nd to top level). The Tree class has been provided to you for this problem.




```
1  def width(root):
2      curr, next = [root] ,[]
3      largest = 0
4      while curr or next:
5          if len(curr) == 0:
6              curr = next
7              next = []
8          largest = max(len(curr), largest)
9          nde = curr.pop()
10         if nde.left:
11             next.append(nde.left)
12         if nde.right:
13             next.append(nde.right)
14     return largest
```

5 Scrheaming at the Top of my Lungs

(a) You are a busy chemist working away in your Schemeboratory. You realized that your list is slightly out of order. You want to write a Scheme function that can swap every pair. For example, inputted the list (1 2 3 4) would result in the list being returned (2 1 4 3)

```
(define (swap_pairs lst)
  (cond ((null? lst) lst)
        ((null? (cdr lst)) lst)
  (else ((cons
      (car
        (cdr lst)
      )
    (cons
      (car lst)
      (swap_pairs (cdr (cdr lst)))
    )
    )))
  )
)
(swap_pairs '(1 2 3 4 5));expect(2 1 4 3 5)
```

(b) Now that you've fixed the ordering of your lists you realized that certain lists have special properties. Specifically, if a list is a palindrome (the list is equivalent if you read it forwards or backwards), you can use it to give you a food of your choice. Being that you are a hungry Schematist, you want to get all the food possible so you want to devise a function that will help you determine if a list is a palindrome. Sadly, you lost some of your Scheme functions in the Anemone Outbreak of 1678. You lost the "equal" function in scheme; however, you managed to salvage the "eq" function. Complete the below function to find if a list is a palindrome given your constraints.

```
(define (is_palindrome lst)
  (define (reverse lst)
    (if (null? lst) lst
    (append (reverse (cdr lst)) (list (car lst)))
    )
  )
    (define rev (reverse lst))
    (define (helper lst1 lst2)
      (cond
        ((null? lst1) #t)
        ((eq? (car lst1) (car lst2))
          (helper (cdr lst1) (cdr lst2)))
        (else #f)
      )
    )
  (helper lst rev)
)


(is_palindrome '(1 2 3 4 5)) ;expect #f
(is_palindrome '(1 2 3 2 1)) ;expect #t
```

6 No Ragrats

You're a CS61A student getting ready to take your final. You decide to take a quick nap before the final- sadly you forgot to wake up and slept your way to 100 years in the future. In the future there are a few phrases that are different. Lucky for you, you can use your knowledge of scheme in order to create a mapping between the future language and your modern day lingo.

(a) For the first part of this question, you need to create a mapping between your modern words and the future words. You will be given 2 scheme lists- the first list is a list of your modern words and the second is a list of the future words. The ith index of the future list maps to the ith index of the modern word list.

```
(define (make-dct future past)
  (if (null? future)
    nil
    (cons (cons (car future) (car past))
          (make-dct (cdr future) (cdr past)))
  )
)
;Create a dictionary with the the 1st element of the ith entry being the
; ith entry of future and the 2nd element of the ith entry being the ith
; entry of the past list. You may expect the lists have the same length
(make-dct '(1 2 3 4 5) '(6 7 8 9 10))
;expect ((1 . 6) (2 . 7) (3 . 8) (4 . 9) (5 . 10))
```

(b) Now we have our mapping. Your goal is given a list of words, return a new list with the words from the future language replaced with the words of your past language.

```
(define (translated-word dct word)
    (cond ((null? dct) word)
          ((eq? (car (car dct)) word) (cdr (car dct)))
    (else (translated-word (cdr dct) word))
  ))

(define (translate dct phrase)
  (map (lambda (x) (translated-word dct x)) phrase)
)
;Translate takes in a dictionary and a list of phrases and returns a list
;with the words from the future replaced with those from the past.
(define dct (make-dct '(1 2 3 4 5) '(6 7 8 9 10)))
(translate dct '(1 3 6 0 2))
'expect (6 8 6 0 7)
```

(c) What is the runtime of your translate function given a dictionary of size m and a phrase of size n? (a) O(1) (b) O(N) (c) O(M+N) **(d)O(MN)** (e) O(2^{MN})

(d) You have decided to rewrite your code in Python because you prefer Python. You will start by rewriting your mapping that you did in part a into Python. You will take in 2 Python lists of the same length and return an object of your choice that will be able to perform mapping similar to parts a and b.

```
def make-dct(future, past):
    """
    Returns some mapping between future and past that you will exploit in
    part e.
    """
    return {future[i] : past[i] for i in range(len(future))}
```

(e) Now that you've done that let's reformat part b in terms of Python. You'll be given a Python list and the mapping between futuristic words and past words that you constructed in part d. Your goal is to return a new list with the words from the futuristic language replaces. You may only write one statement; however, it may not fit on on one line so we have given you 2 lines to write the statement if you need it. YOU MAY NOT USE BUILT IN STRING FUNCTIONS/REPLACE FUNCTIONS.

```
1 def translate(dct, phrase):
2     """
3     >>>translate(<some mapping where 1 maps to 4, 2
4                  maps to 5, and 3 maps to 6>, [4,5,6,7])
5     [4, 6, 5, 7]
6     """
7     return [dct[phrase[i]] if phrase[i] in dct \
8             else phrase[i] for i in range(len(phrase))]
```

7 Get Your Yearly Dose Of SQL Injection

(a) You are an actor working on an interesting movie. There are many famous actors on your set and you want to know how much some of them are being paid. Because you are expert computer scientist on top of being an actor you managed to hack into their Database and steal the salaries of your fellow actors. Your current goal is to find what the 3rd highest salary is (there are at least 3 actors). Your table will be as follows and is named ACTORS:

Actor	Salary
Billy Bob	100
Kartik Bob	1
Kazoo Fazoo	10
Bill Rates	999
Richie Rich	999

We would run our query and the 3rd highest salary would be 10 because Note in the above table because 999 appears twice. In other words, if a salary cannot take up two places (show up in the top 3 more than once) even if more than 1 person has that salary.

```
1 SELECT max(salary)
2       FROM Employee AS e2
3         WHERE (SELECT COUNT(DISTINCT salary)
4               FROM Employee AS e1
5               WHERE e2.salary <= e1.salary) = 3
```

(b) Now you have decided to follow your friends on social media. Before you follow anyone you want to see how many follows they have to see if they are worthy of your follow. You will be given a table of followers and followees. You should output a new table called follower count that has 2 columns, a followee column with a name user (if they have followers) and a num column which has the number of followers they have. Our table will be called FOLLOW. Note that a person cannot follow themselves. Also you should sort the resulting table alphabetically (a - z)

followee	follower
Kazoo Fazoo	Billy Bob
Kartik	Billy Bob
Kartik	Billy Bob
Kazoo Fazoo	Kartik

Calling our Query on the above table will yield the table

followee	num
Kartik	1
Kazoo Fazoo	2

```
1 CREATE TABLE follower_count AS
2 SELECT DISTINCT follower, num FROM follow, (SELECT followee,
3   COUNT(DISTINCT follower)
4 AS num FROM follow GROUP BY followee) AS t WHERE follower = t.followee
5 ORDER BY follower;
```