FINAL REVIEW

COMPUTER SCIENCE MENTORS 61A

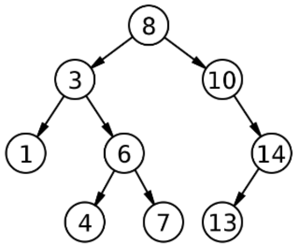
May 3, 2017

1 Binary Trees

1. Given a binary tree t, determine whether or not it is a valid Binary Tree.

```
def valid_bst(t):
  def helper(t, minVal, maxVal):
    if ____________________:
      return True
    check_left = ________________________________
    check_right = _______________________________
    if ___________________________________:
      if check_left and check_right:
        return True
    return False
  return _____________________________________
```

2. Given a Binary Tree, `t` and an integer `k`, return the sum of all entries in the Binary Tree at or within 1 level of the `k`th level. The root is at level 0. You may assume that all entries are integers. Let the `t` in the doctests be the following tree:



```
def approxLevelSum(t, k):
  """
  >>> approxLevelSum(t, 1)
  42  #8 + 3 + 10 + 1 + 6 + 14
  >>> approxLevelSum(t, 2)
  58
  >>> approxLevelSum(t, 3)
  45
  >>> approxLevelSum(t, 0)
  21
  """
  if ____________________________________:

      return 0

  elif ____________________________________:

      return t.label

  elif ____________________________________:

      return ________________________________________________

  else:

      return ________________________________________________
```

2 Sets

1. Given a list of lists of letters, write a solution that will return a list of the letters in every single list in the input. You may assume that the input list contains at least one list.

```
def common_letters(letterLists):
  """
  >>> one, two = ['a', 'b', 'c'], ['d', 'c', 'b']
  >>> three = ['c', 'e']
  >>> common_letters([one, two, three])
  ['c']
  """
  result = __________________________________________

  for ______________________________________________________:

    result = _______________________________________________

  return __________________________________________
```

3 Streams

1. Let's say we were to modify the rest function in the stream class

```
@property
def rest(self):
  if self._compute rest is not None:
    print( computing rest )
    self._rest = self._compute rest()
    self._compute rest = None
  return self._rest
```

What would Python display if we executed the following code?

Expression	Interactive Output

`>>> int_strm = make_integer_stream()`
`>>> print(int_strm.rest)`	

`>>> print(int_strm.rest)`	

2. Takes in a list, `lst` and a positive integer `i` and returns an infinite stream of every `i`th element in `lst`. If the `lst` has no more elements, cycle back to the beginning of `lst` and continue counting.

```
def cycle_ith(lst, i):
  seen = 0

  while seen < i:

    curr = __________

    _________________

    lst.append(curr)

    _________________

  ____________________________________
```

3. Takes in a stream and returns a new stream that contains each element of the input stream only once and in the same order. Assume that the stream passed in is finite.

```
def set_stream(s):
  seen = []

  def compute_rest(curr_stream):

    ______________________________

    ______________________________

    if _____________________________:

      seen.append(curr_stream.first)

      _______________________________________

    else:

      _______________________________________

  return compute_rest(s)
```

4 Scheme

1. What Would Scheme Display?

Expression	Interactive Output
`scm> 'csm`	
`scm> (if 0 1 (/ 1 0))`	
`scm> (or 1 and 2)`	
`scm> (and 1 2)`	
`scm> (or 1 2)`	
`scm> (/ 5)`	
`scm> (* 5)`	
`scm> (- 5)`	
`scm> (+ 5)`	

2. Draw the box and pointer diagrams for the following lists.

```
scm> '(1 2 3)                    scm> '(1 (2 (3 (4))))
```

5 Interpreters

Special form	Rules for evaluation
`begin`	evaluate all expressions
`and`	evaluate expressions until one evaluates to a false-y value
`or`	evaluate expressions until one evaluates to a truth-y value
`cond`	evaluate predicate expressions until one evaluates to a truth-y value, then evaluate the corresponding expression
`if`	evaluate the predicate, then evaluate the 2nd expression if the predicate is truth-y or the 3rd expression if the predicate is false-y
`let`	evaluate expressions in bindings, then evaluate expressions in the body
`define`	no evaluation
`lambda`	no evaluation
`quote`	no evaluation

1. How many calls to `scheme_eval`? How many calls to `scheme_apply`?

Expression	Eval	Apply
`(and 'pizza 'with 'pepperoni #f 'pineapple)`		
`(lambda (cs61a)cs61a)`		
`(if 6 1 (+ 1 0))`		

`(define (gibbesify g)`
`  (cond`
`    ((= g 0)'scheminerverday)`
`    ((= g 1)(gibbesify (- g 1))`
`    (else (gibbesify (- g 2)))`		

`(let ((condvoluted gibbesify))(condvoluted 2))`		

6 Tail Recursion

1. Is it tail recursive?

Expression	Yes/No

`(define (cream cheese)`
`   (if (null? cheese)`
`      0`
`      (+ (car cheese)(cream (cdr cheese)))))`	

`(define (introducing professor dog)`
`   (if (null? professor)`
`      dog`
`      (introducing (cdr professor)(+ (car professor`
`)dog))))`	

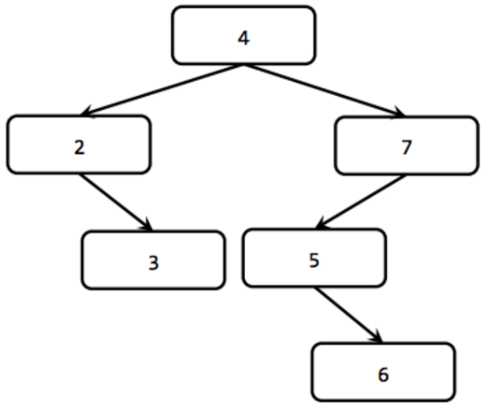
`(define (fact n)`
`   (if (= n 1)`
`      n`
`      (* n (fact (- n 1)))))`	

2. Here are constructors and selectors for a Binary Tree.

```
(define (tree entry left right)       (define (left tree)
  (cons entry (cons left right)))       (car (cdr tree)))

(define (entry tree)                  (define (right tree)
  (car tree))                           (cdr (cdr tree)))
```

This procedure takes a binary search tree, as well as an item contained in the binary search tree. It returns a list of the values encountered along the path from the root to the node containing that item. The `t` in the doctests is defined as follows:



```
(define (bst-path bst item)
```

7 SQL

1. We can use SQL to determine the anagrams of a word! Specifically, lets use SQL to find the anagrams of ’paul’.

```
with
  given (char, weight) as (
    select 'p' , 0001 union
    select 'a' , 0010 union
    select 'u' , 0100 union
    select 'l' , 1000
  )

select ________________________________

_______________________________________

from __________________________________

_______________________________________

where _________________________________;
```

2. After more than 100 years of operation, the Ringling Bros. circus is closing. A victory for animal rights advocates, the circus closure poses a challenge for the zoologists tasked with moving the circus animals to more suitable habitats.

   The zoologists must first take the animals in a freight elevator with a weight limit of 2000. In order to speed-up the process, the zoologists prefer to take groups of animals of the same species in the elevator, rather than one animal at a time.

   Assume the zoologists will only put all of the animals of a particular species in the elevator, or take animals of that particular species one at a time.

   You have access to the table `animals`, with columns containing the animals names, weights, and species.

   Write a query that returns the collective weight and species of animals in a group where there is more than one animal of a particular species in a group, and the collective weight of the animals in the group is less than 2000.

   Your query should yield the following result:

   ```
   229 pig
   1618 tiger
   91 dog
   ```

   ```
   select ______________________________

   from ___________________________

   group by ________________________________

   having
   ________________________________;
   ```

3. Now, suppose we have a table height that has the animals' names and heights. Suppose we want to join the tables animals and height. How many rows will the joined table have?

4. To take the animals to their new habitats, the zoologists load the animals into trucks. The zoologists again want to take the animals in groups of the same species, but one of the trucks has a height limit of 5.0.

   Write a query that returns the maximum height and species of animals in a group where the maximum height is less than 5.0. Your query may yield a species where there is only one animal of that particular species.

   Your query should yield the following result:

```
4.1 pig
4 dog
4.9 zebra
```

```
select ______________________________

from ___________________________

where___________________________

group by ________________________________

having
_______________________________;
```