CS 61A Structure and Interpretation of Computer Programs
Summer 2025

Final Solutions

INSTRUCTIONS

This is your exam. Complete it either at exam.cs61a.org or, if that doesn't work, by emailing course staff with your solutions before the exam deadline.

This exam is intended for the student with email address `<EMAILADDRESS>`. If this is not your email address, notify course staff immediately, as each exam is different. Do not distribute this exam PDF even after the exam ends, as some students may be taking the exam in a different time zone.

For questions with **circular bubbles**, you should select exactly *one* choice.

- ○ You must choose either this option
- ○ Or this one, but not both!

For questions with **square checkboxes**, you may select *multiple* choices.

- ☐ You could select this choice.
- ☐ You could select this one too!

You may start your exam now. Your exam is due at <DEADLINE> Pacific Time. Go to the next page to begin.

Preliminaries

You can complete and submit these questions before the exam starts.

(a) What is your full name?

(b) What is your student ID number?

(c) What is your @berkeley.edu email address?

(d) Sign (or type) your name to confirm that all work on this exam will be your own. The penalty for academic misconduct on an exam is an F in the course.

1. (3.0 points) I Wish I Was Special

(a) (1.0 pt) CS Theory

What does NP stand for?

- ○ Non-computable Problem
- ○ Not Polynomial
- ○ Non-decidable Problem
- [✓ ● Non-deterministic Polynomial]

(b) (1.0 pt) Computational Biology

What is the Central Dogma of Biology?

- ○ DNA is transcribed into proteins, which is then translated into RNA.
- [✓ ● DNA is transcribed into RNA, which is then translated into proteins.]
- ○ DNA is translated into proteins, which is then transcribed into RNA.
- ○ DNA is translated into RNA, which is then transcribed into proteins.

(c) (1.0 pt) AI Safety

What is outer misalignment?

- ○ Model learns wrong goal through looking at data
- ○ Model tricks user into thinking that they achieved their goal
- [✓ ● Whatever goals we specify doesn't match our true intended goals]
- ○ Unequal base rates between groups can lead to disparities in false positives and false negatives

2. (5.0 points) What Would Python Display?

Assume the code below has been executed.

```
def box():
    items = [range(2, 8), ([[3, 5], [7, 11]], lambda s, t: t + s), 10]
    for s in items:
        yield iter(s)

def search(s):
    if next(s) != 10:
        find = list(next(s))
        found = find[1]
        yield from found(find[0][0], find[0][1])
```

Write the output that would be displayed by printing the result of each expression.

- If an iterator object is returned, write ITERATOR.
- If an error occurs, write ERROR.

Each expression below should be evaluated independently of previously-evaluated expressions. For example, a call to `next()` in blank (a) will not affect blank (b).

(a) (1.0 pt) `list(filter(lambda x: x % 2 == 0, next(box())))`

```
[2, 4, 6]
```

(b) (1.0 pt) `next(box())`

ITERATOR

(c) (2.0 pt) `list(search(box()))`

- ○ `ITERATOR`
- ○ `ERROR`
- ○ `[2, 3, 4, 5, 6, 7, 8]`
- ○ `[3, 5, 7, 11]`
- [✓ `[7, 11, 3, 5]`]
- ○ `8`
- ○ `[3, 7]`
- ○ `[5, 11]`

(d) (1.0 pt) `list(search(next(next(box()))))`

ERROR

3. (7.0 points) Balls

Having been inspired by CS 70, Chris and Cedric are playing a game where there are n balls in a bin. The game works as follows:

- Players take alternating turns.
- At every turn, a player can choose `1` to `m` balls to remove from the bin. If there are fewer than `m` balls currently in the bin, players can remove at most the number of current balls in the bin.
- A player wins if they remove the last ball from the bin.
  - i.e., A player loses if it's their turn and there's no more balls (the opponent removed the last ball).

Assuming Chris goes first and Cedric goes second, fill out `num_ways` to count the number of ways Chris can win. In other words, Chris is player 0 and Cedric is player 1.

```
def num_ways(n, m):
    """
    Players take turns removing 1 to m balls. Return the number of ways Chris can win by
    removing the last ball on his turn.

    >>> num_ways(1, 1)
    1
    >>> num_ways(2, 1)
    0
    >>> num_ways(4, 2)
    3
    >>> num_ways(10, 3)
    137
    """
    def helper(player, balls):
        if player == 1 and _______:
                             (a)
            return 1
        if _______:
             (b)
            return 0
        return _______([helper(_______, _______) _______])
                 (c)                (d)      (e)      (f)
    return helper(0, n)
```

(a) **(1.0 pt)** Fill in blank (a).

```
balls == 0
```

(b) **(1.0 pt)** Fill in blank (b).

[✓ `balls <= 0`]

○ `balls == 0`

○ `helper(player, balls - m)`

○ `n < m`

(c) **(1.0 pt)** Fill in blank (c).

- [✓ `sum`]
- `any`
- `all`
- `max`

(d) **(1.0 pt)** Fill in blank (d). **You may not use `if` or `not`.**

```
1 - player
```

(e) **(1.0 pt)** Fill in blank (e).

```
balls - i
```

(f) **(1.0 pt)** Fill in blank (f).

- `for i in range(m)`
- [✓ `for i in range(1, m + 1)`]
- `for i in range(m) if player == 0`
- `for i in range(1, m + 1) if player == 0`

(g) **(1.0 pt)** What is the order of growth of the time it takes to evaluate `bins(n)` in terms of positive integer `n`?

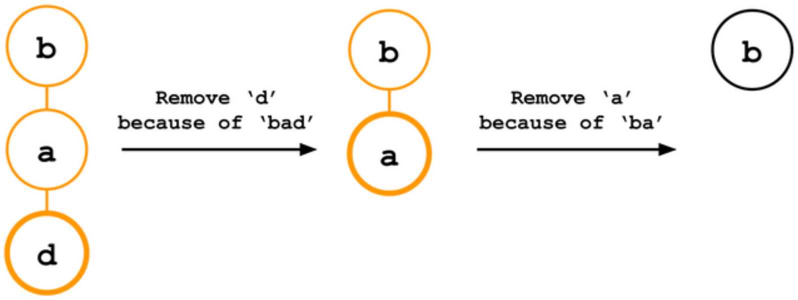
```
def bins(n):
    if n == 1:
        return 1
    if n < 1:
        return 0

    res = 0
    for i in range(1, 5):
        res *= bins(n-i)
    return res
```

- [✓ exponential]
- quadratic
- linear
- logarithmic
- constant

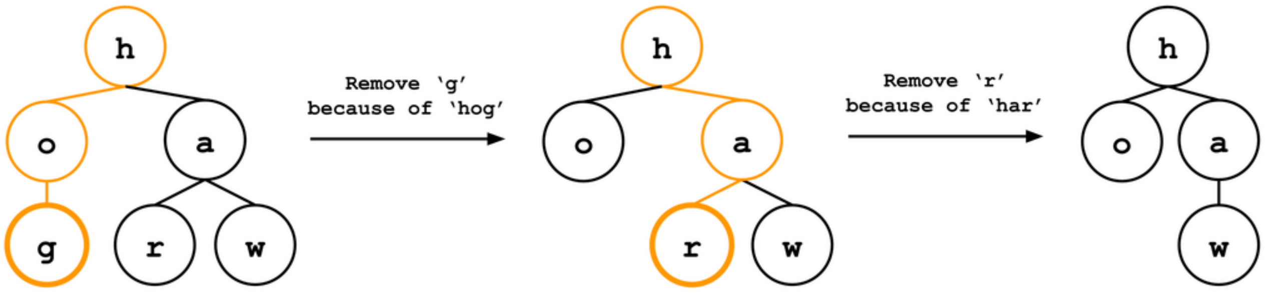
4. **(9.0 points) Valid Triel**

You are given a list of strings, `banned`, and a tree, `t`, in which each node contains a letter, and paths from root to leaf create strings. Remove leaf nodes that cause a path to form a banned string. Your final tree should not contain any paths that form banned strings. Assume all nodes contain valid characters and the root is non-empty and its character is not a banned string.

```
>>> t1 = Tree('b', [Tree('a', [Tree('d')])])
>>> valid_triel(['bad', 'ba'], t1)
>>> t1
Tree('b')
```



```
>>> t2 = Tree('h', [Tree('o', [Tree('g')]), Tree('a', [Tree('w'), Tree('r')])])
>>> valid_triel(['hog', 'how', 'har'], t2)
>>> t2
Tree('h', [Tree('o'), Tree('a', [Tree('w')])])
```



```
def valid_triel(banned, t):
    """
    Mutate the tree by removing leaf nodes such that it contains no banned strings.
    """
    def helper(t, word):
        for b in _______:
                   (a)
            if not _______:
                      (b)
                _______
                  (c)
        if _______ and _______:
             (d)         (e)
            return False
        return True
    helper(t, t.label)
```

(a) (2.0 pt) Fill in blank (a).

```
t.branches[:]
```

(b) (2.0 pt) Fill in blank (b). **You may not use `and` or `or`.**

```
helper(b, word + b.label)
```

(c) (1.0 pt) Fill in blank (c).

- [✓ `t.branches.remove(b)`]
- ○ `t.branches.pop()`
- ○ `helper(b, word)`
- ○ `helper(b, word + b.label)`

(d) (2.0 pt) Fill in blank (d). **Select all that apply.**

- [✓ `t.is_leaf()`]
- ☐ `not t.is_leaf()`
- ☐ `t.branches`
- [✓ `not t.branches`]
- ☐ `t.label in banned`
- ☐ `t.label not in banned`

(e) (2.0 pt) Fill in blank (e).

```
word in banned
```

5. (14.0 points) CS 61A Text Editor

A text editor can be implemented as a Linked List of characters. A character is a string of length 1.

For each subpart, you may assume all the previous subparts have been implemented correctly. For example, when working on part (c), you may assume that parts (a) and (b) have been implemented correctly.

`link_from_program` is implemented for you. `link_from_program` takes in a string, `program`, and returns a Linked List where each element of the Linked List is a character of `program`. Assume it is implemented correctly and you may use it in any subpart of this problem.

```
def link_from_program(program):
    """
    >>> s = link_from_program('print("hello")')
    >>> print(s)
    <p r i n t ( " h e l l o " )>
    """
    # IMPLEMENTATION OMITTED
```

(a) (5.0 points)

Implement `find`, which takes in a Linked List of characters, `s`, and a string `val` and returns the index of the first occurrence of `val` in `s`. If `val` does not exist in `s`, return `float('inf')`. Assume `len(val) >= 1`. You can assume the characters of `val` will appear only consecutively in `s`, or not at all.

```
def find(s, val):
    """Assume len(val) >= 1. Return the index of the first occurrence of val
    >>> s = link_from_program('print("pal")')
    >>> print(s)
    <p r i n t ( " p a l " )>
    >>> find(s, 'pal')
    7
    >>> find(s, 'pals')
    inf
    >>> find(s, 'p')
    0
    >>> find(s, '"p')
    6
    >>> find(s, 'print("pal")')
    0
    """
    if s is Link.empty:
        return float('inf')
    if s.first == val[0]:
        if len(val) == 1:
            return _______
                      (a)
        else:
            return _______(find(s.rest, _______), 1 + find(s.rest, _______))
                      (b)                  (c)                        (d)
    return 1 + _______
                  (e)
```

i. (1.0 pt) Fill in blank (a).

- ○ `-1`
- [✓ `0`]
- ○ `1`
- ○ `val`
- ○ `val[0]`

ii. (1.0 pt) Fill in blank (b).

- [✓ `min`]
- ○ `max`
- ○ `sum`
- ○ `find`
- ○ `len`
- ○ `list`
- ○ `any`
- ○ `all`
- ○ `pow`

iii. (1.0 pt) Fill in blank (c).

```
val[1:]
```

iv. (1.0 pt) Fill in blank (d).

```
val
```

v. (1.0 pt) Fill in blank (e).

```
find(s.rest, val)
```

(b) (3.0 points)

Implement `insert`, which takes in a Linked List of characters, `s`, a string `val`, and an index `i` and mutates `s` such that `val` is inserted as a Linked List of characters in `s` starting at index `i`. `insert` returns `None`. Assume that `i > 0`.

```
def insert(s, val, i):
    """Assume i > 0.

    >>> s = link_from_program('print("pal")')
    >>> insert(s, 'ace', 10)
    >>> print(s)
    <p r i n t ( " p a l a c e " )>
    """
    while i > 1:
        s = s.rest
        i -= 1
    tail = s.rest
    s.rest = _______
                (f)
    while _______ is not Link.empty:
             (g)
        s = s.rest
    _______ = tail
      (h)
```

i. (1.0 pt) Fill in blank (f).

```
link_from_program(val)
```

ii. (1.0 pt) Fill in blank (g).

```
s.rest
```

iii. (1.0 pt) Fill in blank (h).

```
s.rest
```

(c) (2.0 points)

Implement `delete`, which takes in a Linked List of characters, `s`, an index `i`, and a positive integer `n` and mutates `s` such that the first `n` characters of `s` starting at index `i` are removed. `delete` returns `None`. Assume that `i > 0` and `i + n` is less than or equal to length of `s`.

```
def delete(s, i, n):
    """Assume i > 0 and (i + n) is less than or equal to the length of s.

    >>> s = link_from_program('print("pal")')
    >>> delete(s, 7, 3)
    >>> print(s)
    <p r i n t ( " " )>
    """
    while i > 1:
        s = s.rest
        i -= 1
    tail = s.rest
    while n > 0:
        -------
          (i)
        n -= 1
    -------
      (j)
```

i. (1.0 pt) Fill in blank (i).

```
tail = tail.rest
```

ii. (1.0 pt) Fill in blank (j).

```
s.rest = tail
```

(d) (4.0 points)

Implement `find_and_replace`, which takes in a Linked List of characters, `s`, a string, `old`, and a string `new` and mutates `s` such that the first occurrence of the characters of `old` are replaced with the characters of `new`. `find_and_replace` returns `None`. Assume that `old` exists in `s` and the first occurrence of `old` begins at an index greater than 0.

```
def find_and_replace(s, old, new):
    """Assume old exists in s and the first occurrence of old begins at an index
    greater than 0.

    >>> s = link_from_program('x=1+1')
    >>> print(s)
    <x = 1 + 1>
    >>> find_and_replace(s, '1', '4')
    >>> print(s)
    <x = 4 + 1>
    >>> find_and_replace(s, '+1', '-3')
    >>> print(s)
    <x = 4 - 3>
    """
    i = _______
          (k)
    delete(s, i, _______)
                   (l)
    insert(s, _______, _______)
                (m)      (n)
```

i. (1.0 pt) Fill in blank (k).

```
find(s, old)
```

ii. (1.0 pt) Fill in blank (l).

```
len(old)
```

iii. (1.0 pt) Fill in blank (m).

```
new
```

iv. (1.0 pt) Fill in blank (n).

```
i
```

6. (5.0 points) Major Debacle

A prospective transfer student wants to understand applications statistics at UC Berkeley. The `majors` has one row per `major`, and contains its corresponding `discipline` and `college`. The `stats` table has one row per `major`, and includes that major's number of `applicants`, `admits`, and `enrolled` students.

majors:

discipline	college	major
Architecture	Environmental Design	Architecture
Arts and Humanities	Letters & Science	History
Arts and Humanities	Letters & Science	Linguistics
Business Administration	Business	Business Administration
Computer Science	Letters & Science	Computer Science
Computer Science	Letters & Science	Data Science
Engineering	Engineering	Bioengineering
Engineering	Engineering	EECS
Engineering	Engineering	Mechanical Engineering
Life Sciences	Letters & Science	Integrative Biology

stats:

major	applicants	admits	enrolled
Architecture	230	51	45
Bioengineering	129	23	14
Business Administration	2390	118	103
Computer Science	1320	64	52
Data Science	414	68	50
EECS	1326	147	120
History	282	137	49
Integrative Biology	215	77	34
Linguistics	107	61	22
Mechanical Engineering	574	95	65

yield:

college	yields
Engineering	0.7509433962264151
Letters & Science	0.4518950437317784

Adapted from 2023 data provided by the University of California: https://www.universityofcalifornia.edu/about-us/information-center/transfers-major

Create a new table `yield`, which has a column for each `college`, and a column `yields`, which contains the total yield rate of each college. Only include colleges that have greater than 1 major, and only keep majors with an acceptance rate greater than 10%.

Yield rate: enrolled divided by admitted. Acceptance rate: admitted divided by applicants.

```
CREATE TABLE yield AS
    SELECT college, _______ AS yields FROM majors, stats WHERE _______ AND _______ _______;
                      (a)                                        (b)         (c)     (d)
```

(a) (1.0 pt) Fill in blank (a)

SUM(enrolled) / SUM(admitted)

(b) (1.0 pt) Fill in blank (b).

○ majors.college = stats.major

[✓ majors.major = stats.major]

○ majors.discipline = stats.major

○ stats.applicants > stats.admits

(c) (1.0 pt) Fill in blank (c).

admits / applicants > 0.1

(d) (2.0 pt) Fill in blank (d). You may write AND to continue the WHERE clause (but you don't have to). You may also include other clauses such as GROUP BY, ORDER BY, HAVING, and LIMIT (but you don't have to).

GROUP BY college HAVING COUNT(*) > 1

7. (8.0 points) Rotation Situation

Implement `rotate`, which takes in a list `lst` and shifts all elements to the left by `k` indices. For example, `(rotate '(1 2 3 4 5) 2)` returns `(3 4 5 1 2)` as we shift `1` and `2` by two indices to the left, pushing them to the back of the list.

```
;;; Rotate lst by k indices to the left
;;;
;;; scm> (rotate '(1 2 3 4 5) 0)
;;; (1 2 3 4 5)
;;; scm> (rotate '(1 2 3 4 5) 1)
;;; (2 3 4 5 1)
;;; scm> (rotate '(1 2 3 4 5) 2)
;;; (3 4 5 1 2)
;;; scm> (rotate '(1 2 3 4 5) 7)
;;; (3 4 5 1 2) ; should still work for k > (length list)!

(define (rotate lst k)
    (if (= k 0)

        _______
          (a)
        (rotate (_______ _______ _______)  _______)
    )               (b)     (c)     (d)       (e)
)
```

(a) (1.0 pt) Fill in blank (a).

```
lst
```

(b) (1.0 pt) Fill in blank (b).

○ `car`

○ `cdr`

○ `cons`

○ `list`

[✓ `append`]

(c) (1.0 pt) Fill in blank (c).

```
(cdr lst)
```

(d) (1.0 pt) Fill in blank (d).

```
(list (car lst)) -OR- (cons (car lst) nil)
```

(e) **(1.0 pt)** Fill in blank (e).

```
(- k 1)
```

(f) **(1.0 pt)** Is `rotate` tail recursive?

- [✓ Yes, it is tail recursive.]
- ○ No, it is not tail recursive.

(g) **(2.0 pt)** Which expressions are passed to `scheme_eval` when evaluating `(or (- 4 4) 'salutations)`? **Select all that apply.**

- [✓ `(or (- 4 4) 'salutations)`]
- ☐ `or`
- [✓ `(- 4 4)`]
- [✓ `-`]
- [✓ `4`]
- ☐ `'salutations`

8. (11.0 points) Golden

Hunters and Demons are rival idols who battle through song. Each idol stores enemies in the `enemies` dictionary, where keys are idol objects and values are lists of their enemies (also idol objects). When idols sing, they damage all of their enemies (if any). If an enemy's `health` reaches `0`, they're removed from the enemies dictionary (both as keys and values). Hunters have a `takedown` method that reduces their highest-health enemy's health to 0.

Hint: The get method of a dictionary takes two arguments: key and default. If the key is in the dictionary, its value is returned. If not, default is returned. E.g., `{1:2}.get(1, 3)` evaluates to 2, but `{1:2}.get(5, 3)` is 3.

```
class Idol():
    """An idol who sings.
    >>> rumi, mira, jinu, abby = Hunter('Rumi'), Hunter('Mira'), Demon('Jinu'), Demon('Abby')
    >>> rumi
    Rumi
    >>> jinu.add_enemy(rumi); rumi.add_enemy(abby); rumi.add_enemy(jinu); mira.add_enemy(jinu)
    >>> Idol.enemies
    {Jinu: [Rumi], Rumi: [Abby, Jinu], Mira: [Jinu]}
    >>> jinu.sing(); abby.sing();     # Semicolons run both lines, Abby sings without erroring
    Join the Pride.
    Join the Pride.
    >>> rumi.health                                         # Only Jinu damages Rumi by singing
    2
    >>> mira.sing(); rumi.takedown(); Idol.enemies     # Abby removed as he had highest health
    {Jinu: [Rumi], Rumi: [Jinu], Mira: [Jinu]}
    >>> rumi.sing(); mira.sing(); Idol.enemies                           # Jinu ran out of health
    {Rumi: [], Mira: []}
    """
    enemies, health, damage = {}, 3, 1
    def __init__(self, name):
        self.name = name
    def add_enemy(self, enemy):
        Idol.enemies[self] = Idol.enemies[self] + [enemy] if self in Idol.enemies else [enemy]
    def sing(self):
        for target in _______:
                        (a)
            target.reduce_health(self.damage)
    def reduce_health(self, damage):
        self.health -= damage
        if self.health <= 0:
            Idol.enemies.pop(self, None)                         # Removes the idol from keys
            _______ = {_______: [_______] for t in Idol.enemies}  # Removes the idol from values
              (b)        (c)       (d)
    def _______(self):
          (e)
        return self.name

class Hunter(Idol):
    def takedown(self):
        e = max(Idol.enemies[self], key=_______)  # Find the top-health enemy. Assume it exists.
        if e:                                (f)
            e.reduce_health(e.health)
class Demon(Idol):
    def sing(self):
        print("Join the Pride.")
        _______
          (g)
```

(a) **(2.0 pt)** Fill in blank (a).

```
Idol.enemies.get(self, []) -OR- self.enemies.get(self, [])
```

The `.get` is necessary if the idol does not have an enemy. This is why `abby.sing()` does not error.

(b) **(1.0 pt)** Fill in blank (b). **Select all that apply.**

- ☐ `self.enemies`
- ☐ `self.enemies[self]`
- [✓] `Idol.enemies`
- ☐ `Idol.enemies[self]`

`self.enemies` does not work as it would create a new `enemies` object attribute, rather than modifying the class attribute.

(c) **(1.0 pt)** Fill in blank (c).

```
t
```

(d) **(3.0 pt)** Fill in blank (d) by completing the list comprehension.

```
enemy for enemy in Idol.enemies[t] if enemy != self
```

You can use any loop variable. You can also use `self.enemies[t]`. You can also use `is not` instead of `!=`.

(e) **(1.0 pt)** Fill in blank (e).

- ○ `__str__`
- [✓] `__repr__`

(f) **(1.0 pt)** Fill in blank (f).

```
lambda x: x.health
```

(g) **(2.0 pt)** Fill in blank (g). **Select all that apply.**

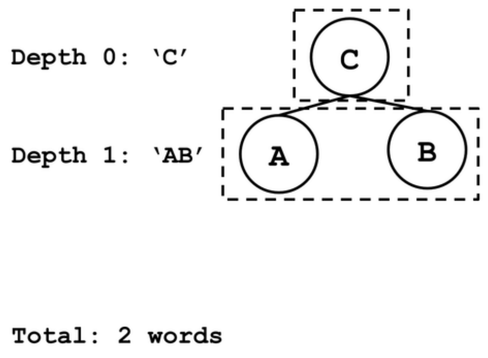
- ☐ `Idol.sing()`
- [✓] `Idol.sing(self)`
- ☐ `Idol(self).sing()`
- ☐ `super().sing(self)`
- ☐ `super(self).sing()`
- [✓] `super().sing()`

9. (8.0 points) Getting Hyped Tonight

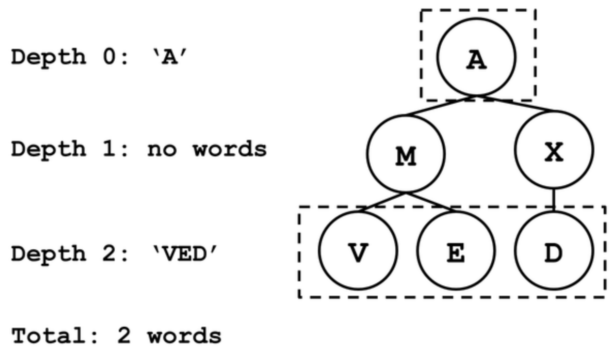
Ved wants to throw a party tonight! Ved has a non-empty tree `t` and a list of strings `hype_strings`. We want to count the number of strings in `hype_strings` that are also in `t`. We say a string is in the tree if the labels of all nodes from left to right at the same depth form the string. Return the number of hype strings in the tree.

Hint: The `pop(index)` method removes and returns the item at the specified index in a list; if no index is given, it removes and returns the last item.

```
>>> t1 = Tree('C', [Tree('A'), Tree('B')])
>>> num_hype(t1, ['AB', 'C'])
2
```



```
>>> t2 = Tree('A', [Tree('M', [Tree('V'), Tree('E')]),
                    Tree('X', [Tree('D')])])
>>> num_hype(t2, ['ME', 'VED', 'A', 'LIT'])
2
```



```
def num_hype(t, hype_strings):
    """
    Return the number of hype_strings that are in the tree.
    """

    nodes_to_visit = [t, None]
    res = 0
    curr_str = ""
    while nodes_to_visit:
        curr_node = _______
                     (a)
        if curr_node is None:   # None represents end of current depth
            if _______:
                (b)

                _______
                  (c)
            curr_str = ""
            if nodes_to_visit:
                nodes_to_visit.append(None)
        else:
            curr_str = _______
                         (d)
            for b in curr_node.branches:
                _______
                  (e)
    return res
```

(a) (2.0 pt) Fill in blank (a).

```
nodes_to_visit.pop(0)
```

(b) (2.0 pt) Fill in blank (b).

```
curr_str in hype_strings
```

(c) (1.0 pt) Fill in blank (c).

- ○ `curr_str += curr_node`
- ○ `nodes_to_visit.append(curr_node)`
- ○ `nodes_to_visit.pop(curr_node)`
- [✓ `res += 1`]

(d) (1.0 pt) Fill in blank (d).

```
curr_str + curr_node.label
```

(e) (2.0 pt) Fill in blank (e).

- ○ `num_hype(b, hype_strings)`
- ○ `num_hype(b, hype_strings[1:])`
- [✓ `nodes_to_visit.append(b)`]
- ○ `nodes_to_visit.extend([b, None])`

10. (5.0 points) Plop, pop, plop

Answer the questions about the code below. Use the free space or scratch paper to draw the diagram to help you answer the questions, however any drawn diagrams will not be graded.

```
def plop(f, lst):
    while s and f(lst, lst.pop()):
        print(lst)
        lst = [f(lst, s.pop())] + lst
    return lst

s = [x + 1 for x in range(1, 4)]
t = plop(lambda t, x: t[len(s) - 1] + x, s)
```

(a) (2.0 pt) What's displayed by executing the above code?

```
[2, 3]
[5]
```

(b) (1.0 pt) How many frames are opened? Assume list comprehensions and method calls like `.pop()` do not open new frames. Do not count the Global frame.

5

(c) (1.0 pt) What would `print(s)` display in the Global frame after executing the above code?

- ○ `[2, 3, 4, 5]`
- ○ `[2, 3, 4]`
- ○ `[3, 4]`
- ○ `[4]`
- [✓ `[]`]
- ○ `[5]`
- ○ `[7]`
- ○ `[5, 7]`
- ○ `[7, 5]`

(d) (1.0 pt) What would `print(t)` display in the Global frame after executing the above code?

- ○ `[2, 3, 4, 5]`
- ○ `[2, 3, 4]`
- ○ `[3, 4]`
- ○ `[4]`
- ○ `[]`
- ○ `[5]`
- ○ `[7]`
- ○ `[5, 7]`
- [✓ `[7, 5]`]

11. (0.0 points) Just for Fun

This is not for points and will not be graded.
Optional: Draw your favorite memory of CS 61A this summer!

We hope you had a wonderful time
learning with us this summer!
♡ SU25 Course Staff

No more questions.