CS 61A Summer 2026

Rebecca Dang

Final

Solutions last updated: Friday, August 14, 2026

Your Name: _____

Your Student ID: _____

Room you are taking the exam in: _____

Name and SID of the person to your **left**: _____

Name and SID of the person to your **right**: _____

You have 170 minutes. There are 10 questions of varying credit. (80 points total)

Question	1	2	3	4	5	6	7	8	9	10	Total
Points	6	7	15	12	10	5	10	5	10	0	80

For questions with **circular bubbles**, you may select only one choice.

- ○ Unselected option (Completely unfilled)
- ◉ Don't do this (it will be graded as incorrect)
- ● Only one selected option (completely filled)

For questions with **square checkboxes**, you may select one or more choices.

- ■ You can select
- ■ multiple squares
- ☑ (Don't do this)

Make sure to **write your student ID at the top of each page** in the provided blank.

Anything you write outside the answer boxes or you ~~cross out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we will grade the worst interpretation. For coding questions with blanks, you may write at most one statement per blank and you may not use more blanks than provided.

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I will follow the rules of this exam. I will not use any other unauthorized materials or devices during this exam. I will not communicate with anyone other than the proctors during this exam. I will not share any information about the exam with anyone until after the end of the exam period.

Acknowledge that you have read and agree to the honor code above and **sign your name below**:

This content is protected and may not be shared, uploaded, or distributed.

Q1 *Haunted Tales* **(6 points)**

For subparts with answer boxes, write exactly what Python would display as output in the interpreter, except in these cases:

1. If a function would be displayed, write `Function`.
2. If an error would be raised, write `Error`.
3. If nothing would be displayed, write `Nothing`.

Clearly indicate whether the output contains single or double quotes (if applicable).

For Q1.1 - Q1.4, consider the following code, which has already been executed (left column top to bottom, then right column top to bottom) with no errors in the same interpreter session.

```
class Spirit:
  boo = 1
  goo = 2

  def __init__(self, boo, bat):
    self.goo = boo
    self.bat = bat

  def __repr__(self):
    return f'Spirit({self.goo}, {self.bat})'
```

```
class Ghost(Spirit):
  def __init__(self, goo, bag):
    super().__init__(goo, bag)

  def __repr__(self):
    return f'Ghost({self.boo}, {self.bat})'

  def __str__(self):
    return 'spooky'

a = Spirit(3, 4)
b = Ghost(5, 6)
```

Q1.1 (1 point) What would be displayed by evaluating the following code?

```
>>> (a.goo, b.goo)
```

```
(3, 5)
```

Q1.2 (0.5 points) What would be displayed by evaluating the following code?

```
>>> b.bag
```

```
Error
```

Q1.3 (0.5 points) What would be displayed by evaluating the following code?

```
>>> b
```

```
Ghost(1, 6)
```

Q1.4 (1 point) What would be displayed by evaluating the following code?

```
>>> Spirit.goo = 7
>>> print(f'{a}, {eval(repr(b))}')
```

```
Spirit(3, 4), spooky
```


This content is protected and may not be shared, uploaded, or distributed.

For Q1.5 - Q1.6, consider the following code which has already been executed with no errors in the same interpreter session. If there are multiple lines of output, write all of them and clearly indicate when a new line begins.

```
class Odyssey(Exception):
  def __init__(self, message, penelope, telemachus):
      super().__init__(message)
      self.penelope = penelope
      self.telemachus = telemachus

def voyage(n):
  if (n // 10) or (n and (5 / 0)):
      raise Odyssey("Ithaca", "Sparta", "Troy")
  else:
      roman_numerals = {'I': 1}
      print(not print(horse(2)) and roman_numerals['V'])

def horse(n):
  try:
      voyage(n)
  except ZeroDivisionError:
      return 'Helen'
  except KeyError:
      print('Circe')
  except Odyssey as ex:
      print(ex.penelope)
      return ex.telemachus
```

Q1.5 (1 point) What would be displayed by evaluating the following code?

```
>>> horse(17)
```

```
Sparta
'Troy'
```

Q1.6 (1 point) What would be displayed by evaluating the following code?

```
>>> horse(0)
```

```
Helen
Circe
```


This content is protected and may not be shared, uploaded, or distributed.

Q1.7 (1 point) Consider a function `find_path`, which takes a `Tree` `t` with unique labels and a value `x`. It returns a list containing the labels of the nodes along a path from the root of `t` to a node labeled `x`. If `x` is not a label in `t`, return `None`.

Below is a **buggy implementation** of `find_path`.

```
1  def find_path(t: Tree, x) -> list:
2    """
3    >>> t1 = Tree(3, [Tree(4), Tree(5, [Tree(6)])])
4    >>> find_path(t1, 5)
5    [3, 5]
6    >>> find_path(t1, 4)
7    [3, 4]
8    >>> find_path(t1, 6)
9    [3, 5, 6]
10   >>> print(find_path(t1, 2))
11   None
12   """
13   if t.label == x:
14       return t.label
15   for b in t.branches:
16       path = find_path(b, x)
17       if path:
18           return [t.label] + path
19   return None
```

When running the doctests, you see the following error message:

```
TypeError: can only concatenate list (not "int") to list
```

What should be changed in order to fix the implementation (ignore indentation)?

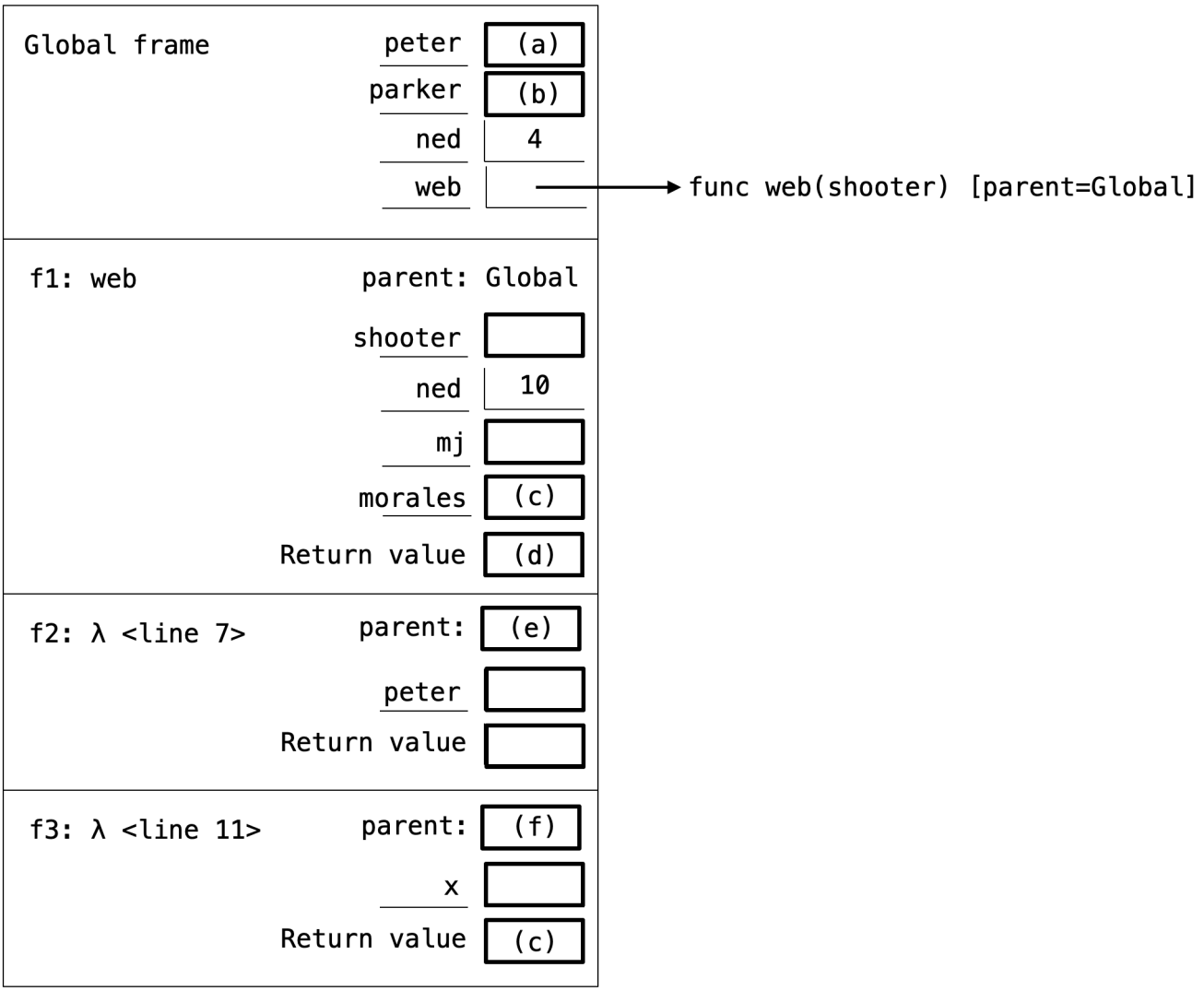
- ○ Change Line 13 to `if t.is_leaf():`
- ● Change Line 14 to `return [t.label]`
- ○ Change Line 16 to `path = find_path(t, x)`
- ○ Change Line 17 to `if path is not None:`
- ○ Change Line 18 to `return [t.label] + [path]`

This content is protected and may not be shared, uploaded, or distributed.

Q2 *Spider-Man: Brand New Environment Diagram* **(7 points)**

The following code is run and produces the shown environment diagram. Assume there are no errors when running this code. Blank boxes in the diagram and your scratch work on the diagram will not be graded.

```
1  peter = [1, 2, 3]
2  parker = [peter, 2]
3  ned = 4
4
5  def web(shooter):
6      ned = 10
7      mj = lambda peter: shooter
8      morales = mj(parker)(parker[0])
9      return peter.pop()
10
11 web(lambda x: (x.pop(0) + peter.pop() + ned))
```



This content is protected and may not be shared, uploaded, or distributed.

For all subparts below, choose or write what goes in each blank **after executing all of the code**.

Q2.1 (1 point) Fill in blank (a).

- ● `[]`
- ○ `[2]`
- ○ `[[], 2]`
- ○ `[2, 3]`
- ○ `[4, 2]`

Q2.2 (1 point) Fill in blank (b).

- ○ `[]`
- ○ `[4, 2]`
- ● `[[], 2]`
- ○ `[[3], 2]`
- ○ `[[2], 2]`

Q2.3 (1 point) Fill in blank (c).

○ `1`	○ `2`	○ `3`	○ `4`
○ `5`	● `8`	○ `10`	○ `14`

Q2.4 (2 points) Fill in blank (d).

```
2
```

Q2.5 (1 point) Fill in blank (e).

- ○ `Global`
- ● `f1`
- ○ `f2`
- ○ `f3`

Q2.6 (1 point) Fill in blank (f).

- ● `Global`
- ○ `f1`
- ○ `f2`
- ○ `f3`

This content is protected and may not be shared, uploaded, or distributed.

Q3 *61A Staff Goes to the Movies!* **(15 points)**

Brian just opened a new movie theatre known as Brian Multi-Cinema (BMC) and needs your help to implement this system! A `Theatre` can show multiple movies at any given time. Each movie can be shown at one or more `Showtime`s (for example, *The Odyssey* could be shown at 12 pm and 3 pm, and *Spider-Man 4* could be shown at 12 pm and 6 pm). A `User` can reserve movie tickets.

```
class Showtime:
  max_capacity = 3

  def __init__(self, movie_name: str, theatre: Theatre, time: str):
    self.movie_name = movie_name
    self.theatre = theatre
    self.time = time
    self.capacity = Showtime.max_capacity

  def __repr__(self):
    return f'{self.movie_name} at {self.time}'

class Theatre:
  def __init__(self):
    # dictionary with key: movie name (str), value: list of Showtime objects
    self.showtimes = {}
    def next_showtime():
      while True:
        yield from ['12 PM', '3 PM', '6 PM', '9 PM']
    self.showtime_generator = next_showtime()

class User:
  max_reservations = 4

  def __init__(self, name: str):
    self.name = name
    self.reservations = []  # list of Showtime objects

  def __repr__(self):
    return self.name
```


This content is protected and may not be shared, uploaded, or distributed.

Q3.1 - Q3.4 (4 points)

Implement the `add_showtime` method of the `Theatre` class, which does the following:

- Creates a new `Showtime` instance with the given `movie_name`, associated with the current `Theatre` instance, and at the next time given by the `showtime_generator` instance attribute
- Updates all relevant instance attributes for the `Theatre` and `Showtime` instances (see doctests)
- Returns the created `Showtime` instance

```
def add_showtime(self, movie_name: str) -> Showtime:
  """
  >>> theatre = Theatre()
  >>> theatre.add_showtime('Minions')
  Minions at 12 PM
  >>> theatre.showtimes
  {'Minions': [Minions at 12 PM]}
  >>> theatre.add_showtime('The Odyssey')
  The Odyssey at 3 PM
  >>> theatre.showtimes
  {'Minions': [Minions at 12 PM], 'The Odyssey': [The Odyssey at 3 PM]}
  >>> theatre.add_showtime('Minions')
  Minions at 6 PM
  >>> theatre.showtimes
  {'Minions': [Minions at 12 PM, Minions at 6 PM],
   'The Odyssey': [The Odyssey at 3 PM]}
  """

  time = next(self.showtime_generator)
                      Q3.1


  new_showtime = Showtime(movie_name, self, time)
                                  Q3.2

  if movie_name in self.showtimes:


    self.showtimes[movie_name].append(new_showtime)
                             Q3.3

  else:


    self.showtimes[movie_name] = [new_showtime]
                          Q3.4

  return new_showtime
```


This content is protected and may not be shared, uploaded, or distributed.

Q3.5 (5 points) Implement the `reserve_ticket` method of the `User` class. It takes in a `Showtime` and returns `True` or `False` depending on whether or not the reservation was successful. A reservation succeeds if the `User` has fewer than the maximum number of reservations (not counting the one they are attempting to make right now) and there is capacity in the showtime. You only need to consider the case where a `User` is reserving a ticket for a `Showtime` for the first time.

You may receive partial credit for your solution. You may not need all lines and you may not use more than the lines provided and each line must contain up to one Python statement. Make sure you clearly indent when necessary.

```
def reserve_ticket(self, showtime: Showtime) -> bool:
  """
  Suppose we have already created a Theatre instance `theatre`
  and there are now the following showtimes:

  >>> theatre.showtimes
  {'The Odyssey': [The Odyssey at 12 PM, The Odyssey at 6 PM],
   'Spider-Man 4': [Spider-Man 4 at 3 PM, Spider-Man 4 at 9 PM,
                    Spider-Man 4 at 12 PM]
  }
  >>> lavanya = User("Lavanya")
  >>> spiderman_showtimes = theatre.showtimes['Spider-Man 4']
  >>> [lavanya.reserve_ticket(s) for s in spiderman_showtimes]
  [True, True, True]
  >>> odyssey_showtimes = theatre.showtimes['The Odyssey']
  >>> [lavanya.reserve_ticket(s) for s in odyssey_showtimes]
  [True, False]
  >>> lavanya.reservations
  [Spider-Man 4 at 3 PM, Spider-Man 4 at 9 PM, Spider-Man 4 at 12 PM,
   The Odyssey at 12 PM]
  >>> all([ showtime.capacity == 2 for showtime in lavanya.reservations ])
  True
  """

  ----------------------------------------------------------------------------

  ----------------------------------------------------------------------------

  ----------------------------------------------------------------------------

  ----------------------------------------------------------------------------

  ----------------------------------------------------------------------------

  ----------------------------------------------------------------------------

  ----------------------------------------------------------------------------
```


This content is protected and may not be shared, uploaded, or distributed.

Solution:

```
def reserve_ticket(self, showtime: Showtime) -> bool:
  if len(self.reservations) < User.max_reservations and showtime.capacity > 0:
    showtime.capacity -= 1
    self.reservations.append(showtime)
    return True
  return False
```

Alternatively, in the `if` statement condition, you could replace `User.max_reservations` with `self.max_reservations`.

This content is protected and may not be shared, uploaded, or distributed.

Q3.6 - Q3.9 (4 points) Implement the `reserve_group_tickets` method of the `User` class. Tickets are reserved for the current `User` and everyone in their group for the `Showtime` with the **largest capacity**. Assume there are never ties for showtimes with the largest capacity. If none of the current showtimes have enough capacity, **add a new showtime** and reserve tickets for it. If the `User` reserving for the group cannot reserve a ticket, **no tickets will be reserved**. Otherwise, the `User` should reserve a ticket for themselves and try to do the same for each person in their `group`. **Assume that `add_showtime` and `reserve_ticket` were implemented correctly.**

```
def reserve_group_tickets(self, group: list[User], theatre: Theatre, movie_name: str):
  """
  Suppose there is a Theatre instance `theatre` with a `Showtime` for Spider-Man 4 at
  12 PM, and now the showtime's capacity is 2 because a User reserved a ticket.

  >>> lavanya = User('Lavanya')
  >>> group = [User('Bill'), User('Alex')]
  >>> lavanya.reserve_group_tickets(group, theatre, 'Spider-Man 4')
  Success: Reserved tickets for Spider-Man 4 at 3 PM for 3 people
  >>> theatre.showtimes
  {'Spider-Man 4': [Spider-Man 4 at 12 PM, Spider-Man 4 at 3 PM]}
  >>> theatre.showtimes['Spider-Man 4'][1].capacity
  0
  >>> lavanya.reservations
  [Spider-Man 4 at 3 PM]
  >>> group[0].reservations
  [Spider-Man 4 at 3 PM]
  >>> group[1].reservations
  [Spider-Man 4 at 3 PM]
  """
  assert 0 < len(group) <= Showtime.max_capacity
  assert movie_name in theatre.showtimes
  assert len(theatre.showtimes[movie_name]) > 0

  showtimes = theatre.showtimes[movie_name]

  best_showtime = max(showtimes, key=lambda showtime: showtime.remaining_capacity)
                                                      Q3.6
  if best_showtime.capacity < len(group) + 1:

    best_showtime = theatre.add_showtime(movie_name)
                                 Q3.7

  if not self.reserve_ticket(best_showtime):
                     Q3.8
    print(f'Error: {self.name} already has the max number of reservations')
  else:
    num_reserved = 1
    for user in group:

      if user.reserve_ticket(best_showtime):
                          Q3.9
        num_reserved += 1
    print(f'Success: Reserved tickets for {best_showtime} for {num_reserved} people')
```


This content is protected and may not be shared, uploaded, or distributed.

Q3.10 (2 points) Suppose Brian wants to be able to cancel a showtime through the system. He plans to do this by adding a `cancel` instance method to the `Theatre` class, which removes the cancelled showtime from the theatre's `showtimes` dictionary. However, there is currently no way to remove that showtime from every user's `reservations` list, if they had already reserved a ticket.

What is the best way to solve this problem so that Brian only needs to call `cancel` to cancel a showtime?

● Add an instance attribute to `Showtime` ○ Add an instance attribute to `User`

In **20 words or less**, describe what this new instance attribute should store. Do NOT provide a justification. Anything you write beyond 20 words will not be graded.

The `User`s who have reserved a ticket to the `Showtime`

This content is protected and may not be shared, uploaded, or distributed.

Q4 *Family Trees* **(12 points)**

Implement `exists_path`, a function that takes in a `Tree` instance `t` and a positive integer `k` greater than 1. It returns whether there exists a root-to-leaf path in `t` whose labels are all multiples of `k`. The input `t` should not be modified. The `Tree` class appears on the final study guide.

```
def exists_path(t: Tree, k: int) -> bool:
  """
  >>> exists_path(Tree(1), 2)
  False
  >>> exists_path(Tree(3), 3)
  True
  >>> t1 = Tree(2, [Tree(4, [Tree(5, [Tree(4)]), Tree(6)])])
  >>> exists_path(t1, 2)
  True
  >>> t2 = Tree(6, [Tree(4, [Tree(7)]), Tree(2)])
  >>> exists_path(t2, 2)
  True
  >>> t2.branches.remove(t2.branches[1])
  >>> exists_path(t2, 2)
  False
  """
  if ___(a)___:
    if ___(b)___:
      return True
    return ___(c)___([___(d)___ for b in t.branches])
  return ___(e)___
```

Q4.1 (1 point) Fill in blank (a).

```
t.label % k == 0
```

Q4.2 (1 point) Fill in blank (b).

```
t.is_leaf()
```

Q4.3 (1 point) Fill in blank (c).

○ `sum` ○ `len` ○ `min` ○ `max` ● `any` ○ `all`

Q4.4 (1 point) Fill in blank (d).

```
exists_path(b, k)
```

Q4.5 (1 point) Fill in blank (e).

```
False
```


This content is protected and may not be shared, uploaded, or distributed.

Definition: In a tree, nodes are **siblings** if they share the same parent node (branches of the same `Tree` instance). Given a positive integer `k`, a node is **lonely** if it is the only sibling that has a label that isn't a multiple of `k` (all other siblings (if any) have labels that are multiples of `k`).

Implement `remove_lonely`, a function that takes in a `Tree` instance `t` and a positive integer `k` greater than 1. It removes all non-root lonely nodes (`t` should be mutated in place).

A lonely node should be removed even if it has no other siblings. **The parent of each remaining node should become its nearest ancestor that was not removed.**

Hint: The `index` method of a list takes a value `x` and returns the index of the first occurrence of `x` in the list. `['a', 'b', 'c'].index('b')` evaluates to `1`. If `x` does not occur in the list, it will error.

```
def remove_lonely(t: Tree, k: int) -> None:
  """
  >>> t1 = Tree(3)
  >>> remove_lonely(t1, 3)
  >>> t1  # Nothing happens to root
  Tree(3)
  >>> t2 = Tree(1, [Tree(3, [Tree(4)]), Tree(5), Tree(6)])
  >>> remove_lonely(t2, 2)
  >>> t2  # There are no lonely siblings
  Tree(1, [Tree(3, [Tree(4)]), Tree(5), Tree(6)])
  >>> t3 = Tree(1, [Tree(3, [Tree(8)]),
  ...               Tree(5, [Tree(6), Tree(4), Tree(12)]),
  ...               Tree(9)])
  >>> remove_lonely(t3, 3)
  >>> print(t3)  # There were lonely siblings
  1
    3
    6
    12
    9
  """
  for b in t.branches:
    ___(f)___

  non_multiples = list(filter(lambda b: ___(g)___, t.branches))

  if ___(h)___:   # Is there a lonely sibling?
    to_remove = ___(i)___
    i = t.branches.index(to_remove)
    ___(j)___ = t.branches[:i] + ___(k)___ + t.branches[i+1:]

```

Answer blanks are on the next page.

This content is protected and may not be shared, uploaded, or distributed.

Q4.6 (1 point) Fill in blank (f).

```
remove_lonely(b, k)
```

Q4.7 (1 point) Fill in blank (g).

```
b.label % k != 0
```

Q4.8 (1 point) Fill in blank (h).

- ○ `non_multiples`
- ○ `t.branches`
- ● `len(non_multiples) == 1`
- ○ `len(t.branches) == 1`

Q4.9 (1.5 points) Fill in blank (i).

```
non_multiples[0]
```

Q4.10 (1 point) Fill in blank (j).

```
t.branches
```

Q4.11 (1.5 points) Fill in blank (k).

```
to_remove.branches
```

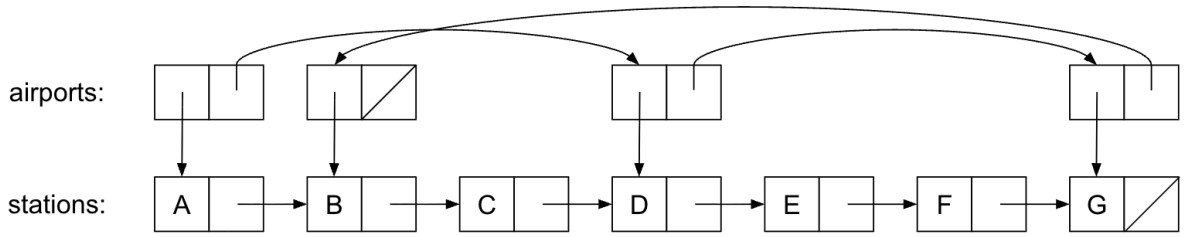

This content is protected and may not be shared, uploaded, or distributed.

Q5 *Commuting to Linked List Land* **(10 points)**

An **airport** is a `Link` whose `first` attribute is a station and `rest` attribute is another airport. A **station** is a `Link` whose `first` attribute is the city name where the station is located (`str`) and `rest` attribute is another station. Implement `best_route`, which returns the **length of the shortest valid travel route** from the first airport in `airports` to the destination city named `dest`. You may assume that `dest` exists, city names are unique, and neither `airports` nor `stations` contain any cycles.

A **valid travel route** starts at the first airport in the `airports` linked list and ends at the destination's train station. At each airport, you can either transfer to the city's train station or take a flight to the next airport. Once you transfer to a station, you cannot return to an airport. Transfers take 1 unit of length, except transferring from a city's airport to its train station, which takes 0 units.

Consider the diagram below, which represents the `airports` and `stations` in the doctests for this problem.



In the doctests below, calling the `setup` function constructs the previous diagram. You do not need to know how it works. Also, for all subparts, do **NOT** violate the abstraction barrier of `Link`.

Hints: `float("inf")` represents a positive infinite value. For all integers `n`, `n < float("inf")` and `n + float("inf") == float("inf")`. Also, the `Link` class appears on the final study guide.

```
def best_route(airports: Link, dest: str) -> int:
    """
    >>> airports = setup(["A", "B", "C", "D", "E", "F", "G"], ["A", "D", "G", "B"])
    >>> best_route(airports, "A")
    0
    >>> best_route(airports, "C") # A -> B -> C
    2
    >>> best_route(airports, "F") # A -> D -> E -> F
    3
    """
    def take_train(stations):
        if ___(a)___:
            return float('inf')
        if ___(b)___:
            return ___(c)___
        return ___(d)___

    if ___(e)___:
        return float('inf')
    return min(___(f)___, ___(g)___)
```


This content is protected and may not be shared, uploaded, or distributed.

Q5.1 (1 point) Fill in blank (a).

- ◯ `stations == Link.empty`
- ◯ `stations == ()`
- ◯ `not stations`
- ● `stations is Link.empty`
- ◯ `stations is ()`

Q5.2 (1 point) Fill in blank (b).

```
stations.first == dest
```

Solution: Alternatively, `stations.first != dest` (see 5.3 and 5.4)

Q5.3 (1.5 points) Fill in blank (c).

```
0
```

Solution: Alternatively, if 5.2 is `stations.first != dest`, this blank should be `1 + take_train(stations.rest)`

Q5.4 (1.5 points) Fill in blank (d).

```
1 + take_train(stations.rest)
```

Solution: Alternatively, if 5.2 is `stations.first != dest`, this blank should be `0`

Q5.5 (1 point) Fill in blank (e).

```
airports is Link.empty
```

Q5.6 (3 points) Fill in blank (f).

```
1 + best_route(airports.rest, dest)
```

Q5.7 (1 point) Fill in blank (g).

- ◯ `take_train(airports)`
- ● `take_train(airports.first)`
- ◯ `take_train(airports.rest)`

This content is protected and may not be shared, uploaded, or distributed.

◯ `1 + take_train(airports)`

◯ `1 + take_train(airports.first)`

◯ `1 + take_train(airports.rest)`

This content is protected and may not be shared, uploaded, or distributed.

Q6 *World Cup Efficiency* **(5 points)**

Q6.1 (2 points)

```
def scoring_ways(n):
  if n == 0:
    return 1
  elif n < 0:
    return 0
  else:
    return scoring_ways(n - 1) + scoring_ways(n - 2)
```

What is the code's order of growth with respect to `n`, an integer?

○ Constant ○ Linear ● Exponential

○ Logarithmic ○ Quadratic

Solution: Most non-base-case calls create two additional recursive calls. The resulting recursion tree grows exponentially with `n`.

Q6.2 (1.5 points)

```
def tournament_data(teams):
  result = []
  for team in teams:
    result.append(team)
  for match_number in range(104):
    result.append(match_number)
  return result
```

What is the code's order of growth with respect to the number of elements in `teams`, an iterable? Assume that `append` takes constant time.

○ Constant ● Linear ○ Exponential

○ Logarithmic ○ Quadratic

Solution: If `teams` contains `n` elements, the first loop executes `n` times. The second loop always executes exactly 104 times. The total work is `n + 104`, which grows linearly with `n`.

Q6.3 (1.5 points)

```
def possible_matchups(teams):
  for team1 in teams:
    for team2 in teams:
      if team1 != team2:
        print(team1, team2)
```

What is the code's order of growth with respect to the number of elements in `teams`, an iterable?

This content is protected and may not be shared, uploaded, or distributed.

○ Constant ○ Linear ○ Exponential

○ Logarithmic ● Quadratic

Solution: The function processes every distinct pair of teams once. If there are `n` teams, the number of pairs is proportional to `n * n`, so the runtime grows quadratically.

Q6.4 This subpart is worth up to **2 points of extra credit**. You must get the entire problem correct to get 2 points of EC, otherwise you will receive 0 points. You should do the rest of the exam first and then come back to this.

Write `count_duplicate_evens`, a function that takes **linear time** ($O(n)$) and returns the number of **even** integers that appear more than once in `scores` (a list of integers).

You may not need all lines and you may not use more than the lines provided. Each line must contain up to one Python statement. Make sure you clearly indent when necessary.

This content is protected and may not be shared, uploaded, or distributed.

```
def count_duplicate_evens(scores: list[int]) -> int:
    """
    >>> count_duplicate_evens([1, 5, 8, 8, 12])
    1
    >>> count_duplicate_evens([4, 3])
    0
    >>> count_duplicate_evens([1, 1, 4])
    0
    >>> count_duplicate_evens([2, 2, 2, 1, 4, 4])
    2
    """

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------
```


This content is protected and may not be shared, uploaded, or distributed.

Solution:

```
def count_duplicate_evens(scores: list[int]) -> int:
  counts = {}
  for score in scores:
    if score % 2 == 0:
      if score not in counts:
        counts[score] = 1
      else:
        counts[score] += 1
  num_duplicates = 0
  for score in counts:
    if counts[score] > 1:
      num_duplicates += 1
  return num_duplicates
```


This content is protected and may not be shared, uploaded, or distributed.

Q7 *It's Time(s) to Bring Multiply Back* **(10 points)**

You are working on the Scheme project while watching a heated pickleball match between global rivals, Erling and Haaland, when suddenly Haaland falls on your computer, causing a glitch that eliminates the `*` procedure.

IMPORTANT: Answer ALL parts without calling the built-in multiplication procedure (i.e. do NOT use `*`).

(a) (3 points) Implement `multiply-ab`, which multiplies integers `a` and `b`.

Hint: `(- 3)` evaluates to `-3`.

```
(expect (multiply-ab 6 7) 42)         ; 42 = 6 + 6 + 6 + 6 + 6 + 6 + 6
(expect (multiply-ab (- 6) (- 7)) 42) ; 42 = - ( -6 + -6 + -6 + -6 + -6 + -6 + -6)
(expect (multiply-ab 6 (- 7)) -42)

(define (multiply-ab a b)
  (cond ((< b 0) (- ___(i)___))
        ((= b 0) 0)
        (else (___(ii)___ a (multiply-ab a ___(iii)___)))))
```

Q7.1 (1 point) Fill in blank (i).

```
(multiply-ab a (- b))
```

Solution: Alternate solutions include:

- `(+ a (multiply-ab a (- (- b) 1)))`
- `(multiply-ab a (+ b 1)) a`

Q7.2 (1 point) Fill in blank (ii).

● `+` ○ `-` ○ `/` ○ `multiply-ab`

Q7.3 (1 point) Fill in blank (iii).

```
(- b 1)
```

(b) (3 points) Implement `multiply-lst`, which multiplies a non-empty list of numbers together.

Hints: You may use the `multiply-ab` function from part (a); assume it works correctly.

```
(expect (multiply-lst '(5 12 6)) 360)
(expect (multiply-lst '(9)) 9)
(expect (multiply-lst '(1 1 1 1)) 1)

(define (multiply-lst lst) (___(i)___ ___(ii)___ ___(iii)___))
```

Q7.4 (1 point) Fill in blank (i).

○ `map` ○ `filter` ● `reduce`

Q7.5 (1 point) Fill in blank (ii).

This content is protected and may not be shared, uploaded, or distributed.

```
multiply-ab
```

Q7.6 (1 point) Fill in blank (iii).

```
lst
```


This content is protected and may not be shared, uploaded, or distributed.

(c) (4 points) Implement `multiply-expr`, which takes an expression `expr` that is either a number OR a valid call to the original `*` procedure. It returns the value of `expr` under a regular Scheme interpreter.

Hint: You may use the `multiply-lst` function from part (b); assume it works correctly.

```
(expect (multiply-expr 9) 9)
(expect (multiply-expr '(* 2 3)) 6)
(expect (multiply-expr '(* (* 1 2) (* 3 (* 4 1 1) 2))) 48)

(define (multiply-expr expr)
  (if (number? expr)
      ___(i)___
      (multiply-lst
        (___(ii)___ ___(iii)___ (___(iv)___ expr))
      )
  )
)
```

Q7.7 (1 point) Fill in blank (i).

`expr`

Q7.8 (1 point) Fill in blank (ii).

- ○ `append`
- ● `map`
- ○ `filter`
- ○ `reduce`

Q7.9 (1 point) Fill in blank (iii).

- ○ `car`
- ○ `cdr`
- ● `multiply-expr`
- ○ `multiply-lst`

Q7.10 (1 point) Fill in blank (iv).

- ○ `car`
- ● `cdr`
- ○ `multiply-expr`
- ○ `multiply-lst`

This content is protected and may not be shared, uploaded, or distributed.

Q8 *Bagel with a Side of Macros* **(5 points)**

For **Q8.1 - Q8.2**, write or select exactly what the Scheme interpreter would display when evaluating the expressions, except in these cases:

1. If a function would be displayed, write `Function`.
2. If an error would be raised, write `Error`.
3. If nothing would be displayed, write `Nothing`.

Q8.1 (1 point)

```
scm> (define x 10)
x
scm> (quasiquote (,x (cons ,3 ,(cons 5 nil))) )
```

```
(10 (cons 3 (5)))
```

Q8.2 (1 point)

```
scm> (define-macro (gold-bean-sandwich a b)
       (append (cons (car a) nil) (cdr b)))
gold-bean-sandwich
scm> (gold-bean-sandwich (+ 1 2) (* 10 5))
```

○ 2 ○ 3 ● 15 ○ 50 ○ Error

Implement the procedure `gold-bean-adder`, which takes in two "lists" of length 1 that each stores a number and returns the sum of the stored numbers. Assume the two arguments are always given in **direct list form**, e.g. `(2)`, not as `(cons 2 nil)` or any similar expressions that evaluate to lists (see doctests for more details).

```
(expect (gold-bean-adder (2) (3))  5) ; 2 + 3
(expect (gold-bean-adder (5) (6)) 11) ; 5 + 6

; If implemented correctly, the following expressions should error:
(gold-bean-adder (cons 5 nil) (cons 6 nil))  ; can't take in cons lists
(gold-bean-adder '(5) '(6))                  ; can't take in quoted lists

(define-macro (gold-bean-adder a b)
  (___(i)___ (___(ii)___  a) (___(iii)___  b)))
```

Q8.3 (1 point) Which of the following could fill in blank (i)? Select all that apply.

■ + ☐ '+ ☐ list + ■ list '+

Q8.4 (1 point) Fill in blank (ii).

```
car
```

Q8.5 (1 point) Fill in blank (iii).

```
car
```


This content is protected and may not be shared, uploaded, or distributed.

Q9 *2026 Formula 1 Season meets SQL* **(10 points)**

You are given some data about the 2026 Formula 1 season so far.

The `drivers` table has one row per driver and the columns: `id` (a unique 3-letter abbreviation), their `nationality` (`string`, a 3-letter abbreviation), their `team` (`string`), and their `points` (`int`). **Each F1 team has two drivers.** There are twenty two total drivers.

The `races` table has one row per race and columns for the `date` (`string`), `winner` (`string`, corresponds to `drivers.id`), and `laps` completed (`int`).

In the tables below, values in string columns are displayed without quotation marks for brevity.

NOTE: For all subparts, remember that a single line of SQL code can contain multiple clauses.

id	nationality	team	points
ANT	ITA	Mercedes	204
RUS	GBR	Mercedes	154
HAM	GBR	Ferrari	159
LEC	MON	Ferrari	126
NOR	GBR	McLaren	103
PIA	AUS	McLaren	92
VER	NED	Red Bull Racing	91
HAD	FRA	Red Bull Racing	60
LAW	NZL	Racing Bulls	39
LIN	GBR	Racing Bulls	22
GAS	FRA	Alpine	42
COL	ARG	Alpine	19
BEA	GBR	Haas F1 Team	18
OCO	FRA	Haas F1 Team	3
BOR	BRA	Audi	10
HUL	GER	Audi	0
SAI	ESP	Williams	6
ALB	THA	Williams	5
BOT	FIN	Cadillac	0
PER	MEX	Cadillac	0
ALO	ESP	Aston Martin	1
STR	CAN	Aston Martin	0

Table 1: `drivers` table

date	winner	laps
08 March	RUS	58
15 March	ANT	56
29 March	ANT	53
03 May	ANT	57
24 May	ANT	68
07 June	ANT	78
14 June	HAM	66
28 June	RUS	71
05 July	LEC	52
19 July	ANT	44

Table 2: `races` table

This content is protected and may not be shared, uploaded, or distributed.

Q9.1 (2 points) You run the following query.

```
SELECT team, COUNT(*) AS count
  FROM drivers AS d, races AS r
  WHERE d.id = r.winner
  GROUP BY team
  ORDER BY COUNT(*) DESC;
```

Complete the table this query returns. The column names are provided; only fill in the result rows. You may not need to use all the rows. You do not need to enclose strings in quotes.

team	count

Solution:

team	count
Mercedes	8
Ferrari	2

Create a table with every driver's ID and their absolute point difference to their teammate. Only include drivers who have more points than their teammate, **except in case of a tie (no point difference)**. In case of a tie, only include the driver whose three-letter abbreviation comes first in the alphabet. Your final table should be ordered by the point difference in descending order, and then by the driver's ID in ascending order.

```
SELECT ___(a)___ as id, ___(b)___ AS differences
FROM ___(c)___ AS a JOIN ___(d)___ AS b
ON ___(e)___ AND ( a.points > b.points OR ( ___(f)___ ) )
ORDER BY ___(g)___;
```

Q9.2 (0.5 points) Fill in blank (a).

- ● `a.id`
- ○ `a.team`
- ○ `a.winner`
- ○ `a.nationality`
- ○ `a.date`
- ○ `a.laps`

Q9.3 (0.5 points) Fill in blank (b).

This content is protected and may not be shared, uploaded, or distributed.

```
a.points - b.points
```

Solution: Note that it is not necessary to use `ABS` if the join condition checks that `a.points >= b.points`.

Q9.4 (0.5 points) Fill in blank (c).

● `drivers` ○ `races`

Q9.5 (0.5 points) Fill in blank (d).

● `drivers` ○ `races`

This content is protected and may not be shared, uploaded, or distributed.

Q9.6 (1 point) Fill in blank (e).

```
a.team = b.team
```

Q9.7 (1 point) Fill in blank (f).

```
a.points = b.points AND a.id < b.id
```

Q9.8 (1 point) Fill in blank (g).

```
differences DESC, a.id
```

Solution: Alternate solution (explicitly add `ASC` keyword): `differences DESC, a.id ASC`. Additionally, it is acceptable to replace `differences` with `a.points - b.points` or `ABS(a.points - b.points)`. Additionally, it is acceptable to replace `a.id` with `id`.

Complete the query so that it returns each team and its total points, but only includes teams with more than 100 total points.

```
1 SELECT team, ___(a)___ AS total_points
2 FROM drivers
3 ___(b)___
4 ORDER BY total_points DESC;
```

Q9.9 (1 point) Fill in blank (a).

- ○ `COUNT(DISTINCT points)`
- ○ `COUNT(*)`
- ○ `MAX(points)`
- ○ `MIN(points)`
- ● `SUM(points)`
- ○ `points`

Q9.10 (2 points) Fill in blank (b).

```
GROUP BY team HAVING SUM(points) > 100
```

Solution: In SQLite, it is acceptable to replace `SUM(points) > 100` with `total_points > 100`. (This is not the case for most other SQL variants though due to the order of execution of `SELECT` being after `HAVING`.)

This content is protected and may not be shared, uploaded, or distributed.

SID: _____

Q10 *Just for fun!* **(0 points)**

Q10.1 **You can get up to 1 point of extra credit for answering *at least one* of these questions about the special topics lectures correctly.**

AI Coding Tools: Which of the following model types is the most expensive (hence the most rare)?

○ Open weights ○ Open source ○ Open door ● Open corpus

Web Applications: What allows you to request and send data between websites/organizations?

● API ○ DAO ○ HTML ○ CSS

Computer Security: Which security principle suggests using multiple parties so that no single point of trust/failure can comprimise the entire system?

○ Ensure Complete Mediation

● Separation of Responsibility

○ Least Privilege

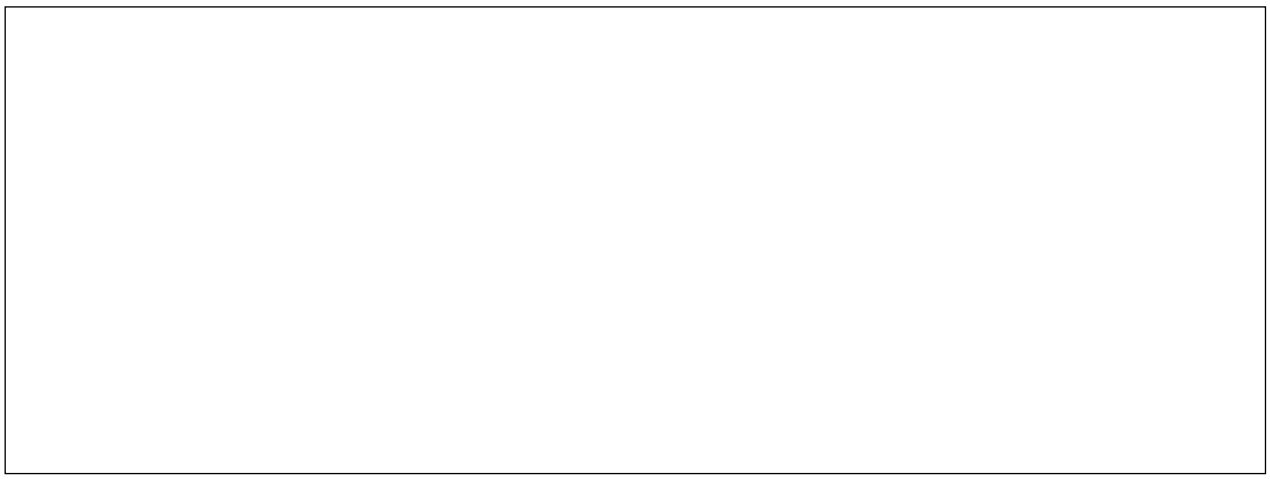
○ Defense in Depth

Q10.2 Here's a quick question just for fun! **There are no points, extra credit, or actual money** for this question, and we'll release the results when we release grades!

You are placed in front of two buttons. Of the students who choose to press a button, if more than 50% of students press the blue button, everyone gets $1 million. If less than 50% of students press the blue button, only the students who press the red button receive $1 million. Which button do you press?

○ Red Button ○ Blue Button

Q10.3 Draw something fun, or write a message for the staff! This is also the place where you can report any potential academic misconduct you witnessed during the exam (please include as much detail as possible). All reports will be kept anonymous.



This content is protected and may not be shared, uploaded, or distributed.