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?

Oski Bear

(b) What is your student ID number?

123456789

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

oski@berkeley.edu

(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.

Oski the Bear
1. (11.0 points)  What Would You Display

(a) (5.0 points)  WWPD

For each expression below, choose the correct option or write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines.

- If an error occurs, write “Error”, but include all output displayed before the error.
- If evaluation would run forever, write “Forever”.
- To display a function value, write “Function”.
- If the evaluated expression wouldn’t display anything, write “Nothing”. The interactive interpreter displays the value of a successfully evaluated expression, unless it is None.
- If you need to write a level of indentation, please draw an arrow, like ->, to represent one singular level of indentation. Draw one arrow for each level of indentation you want to indicate. This may be a tab (\t) or multiple whitespaces.
- Assume the expressions are evaluated in order in the same interactive session, and so evaluating an earlier expression may affect the result of a later one.
- Assume the Tree class has been defined (as written on the Study Guide provided)
- Assume the Tree data abstraction has been defined (as written on the Study Guide provided)

i. (0.5 pt)

```python
>>> print("luck!", print("good"))
```

ii. (1.0 pt)

```python
>>> t = Tree("hi", [Tree("you"), Tree("there")])
>>> print(t)
```

iii. (1.0 pt)

```python
>>> t
```
iv. (1.0 pt)

```python
>>> t.branches.append(Tree("hey", ["hello"]))
```

v. (0.5 pt)

```python
>>> t.label = "hello"
>>> t.branches[0].label = "friend"
```

vi. (1.0 pt) Do any of the above expressions in this question break the abstraction barrier?

- [ ] YES
- [ ] NO
(b) (3.0 points) WWSD

For each expression below, choose the correct option or write the output displayed by the interactive Scheme interpreter when the expression is evaluated. The output may have multiple lines.

- If an error occurs, write “Error”, but include all output displayed before the error.
- If evaluation would run forever, write “Forever”.
- To display a function value, write “Function”.
- If the evaluated expression wouldn’t display anything, write “Nothing”. The interactive interpreter displays the value of a successfully evaluated expression, unless it is undefined.
- Assume the Tree data abstraction has been defined (as written on the Study Guide provided)

i. (0.5 pt)

```
scm> (define t (tree 4 (list (tree 6 nil) (tree 2 (list (tree 7 nil)))))))
```

ii. (1.0 pt)

```
scm> t
```

- (4 6 2 7)
- (4 (6 2 (7)))
- (4 (6) (2 (7)))
- (tree 4 (list (tree 6 nil) (tree 2 (list (tree 7 nil))))))
- (tree 4 (tree 6 nil) (tree 2 (tree 7 nil)))
- Error
- Forever
- Function
- Nothing
iii. (1.0 pt)

```scheme
(define f (lambda (x) (if (null? (cdr x)) (car x) (+ (car x) (f (cdr x))))))
f
(f '(1 2 3))
6
```

Given this error, select all expressions that could have been the input to the interactive Scheme interpreter.

Traceback (most recent call last):
  0 (+ (car x) (f (cdr x)))
Error: operand 0 ((quote hi)) is not a number

- (f '(hi))
- (f '(hi))
- (f (hi))
- (f (quote ((quote hi)))))
- (f '('hi 'hello))
- (f '('hi 2))
- (f '2 'hi))
- (f (2 'hi))

iv. (0.5 pt)

```scheme
(or #f 0 2 (and 3 1 (or 5)))
```
(c) (2.0 points)   WWRD

For each RegEx expression below, choose all options that would fully match if put as test strings in Regex101.

i. (2.0 pt)

`^b[^b-z]{1}[arken]+(bi)*es?$`

- bakes
- baarkenbie
- barre
- bbarbies
- barbie
- ken
- barbiess
- bakbie
(d) (1.0 points) WWSQLD

Assume the records table that you’ve seen in lecture, discussion, and labs so far has been implemented in your interactive SQL interpreter so that it is this table:

<table>
<thead>
<tr>
<th>name</th>
<th>division</th>
<th>title</th>
<th>salary</th>
<th>supervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alyssa P Hacker</td>
<td>Computer</td>
<td>Programmer</td>
<td>40000</td>
<td>Ben Bitdiddle</td>
</tr>
<tr>
<td>Ben Bitdiddle</td>
<td>Computer</td>
<td>Wizard</td>
<td>60000</td>
<td>Oliver Warbucks</td>
</tr>
<tr>
<td>Cy D Fect</td>
<td>Computer</td>
<td>Programmer</td>
<td>35000</td>
<td>Ben Bitdiddle</td>
</tr>
<tr>
<td>Eben Scrooge</td>
<td>Accounting</td>
<td>Chief Accountant</td>
<td>75000</td>
<td>Oliver Warbucks</td>
</tr>
<tr>
<td>Lana Lambda</td>
<td>Administration</td>
<td>Executive Director</td>
<td>610000</td>
<td>Lana Lambda</td>
</tr>
<tr>
<td>Lem E Tweakit</td>
<td>Computer</td>
<td>Technician</td>
<td>25000</td>
<td>Ben Bitdiddle</td>
</tr>
<tr>
<td>Louis Reasoner</td>
<td>Computer</td>
<td>Programmer Trainee</td>
<td>30000</td>
<td>Alyssa P Hacker</td>
</tr>
<tr>
<td>Oliver Warbucks</td>
<td>Administration</td>
<td>Big Wheel</td>
<td>150000</td>
<td>Oliver Warbucks</td>
</tr>
<tr>
<td>Robert Cratchet</td>
<td>Accounting</td>
<td>Scrivener</td>
<td>18000</td>
<td>Eben Scrooge</td>
</tr>
</tbody>
</table>

i. (1 pt) Given this SQL query:

```sql
sql> SELECT AVG(a.salary) FROM records AS a, records AS b WHERE a.name = b.supervisor
    ...> GROUP BY a.division ORDER BY a.division LIMIT 2;
```

which outputs the table below, what is being displayed and represented by this new table? In other words, translate this SQL query into English pseudocode. Please summarize in 1-2 sentences. Responses with more than 2 sentences will be docked points.

<table>
<thead>
<tr>
<th>AVG(a.salary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>75000</td>
</tr>
<tr>
<td>265000</td>
</tr>
</tbody>
</table>
2. (12.0 points) Slipknot

(a) (2.0 points) Product

Implement `product`, a function that takes in a linked list `s` and returns the product over all the values in the linked list. Assume that the initial list `s` is not `Link.empty`.

```python
def product(s):
    ""
    Takes in a linked list S and returns the product over all the values in the linked list.
    >>> product(Link(2, Link(5))) # 2 * 5 = 10
    10
    >>> product(Link(7, Link(2, Link(7)))) # 7 * 2 * 7 = 98
    98
    >>> product(Link(5))
    5
    ""
    if s is Link.empty:
        return _________
    (a)
    return _________
    (b)
```

i. (0.5 pt) Fill in blank (a)

ii. (1.5 pt) Fill in blank (b)
(b) (5.0 points) Untangle

Implement `untangle`, a function that takes in a linked list `s` with strictly positive values. The function `mutates` the linked list to only contain 1s, where the length of the resulting linked list is equal to the sum of the values in the original list. `untangle` potentially modifies the values in `s` and adds new Links to `s`.

For example, calling `untangle` on `<3 1 2>` will mutate the linked list to become `<1 1 1 1 1 1 1>`. See the doctests for more examples.

```python
def untangle(s):
    
    Takes in a linked list `S` and mutates `S` to only contain 1s.
    The length of the resulting list is equal to the sum of the values in the original list. Assume the values in `S` are strictly positive.

    >>> a = Link(5)
    >>> untangle(a)
    >>> print(a)
    <1 1 1 1 1>
    >>> b = Link(3, Link(2, Link(4)))
    >>> untangle(b)
    >>> print(b)
    <1 1 1 1 1 1 1 1 1>
    >>> c = Link(1)
    >>> untangle(c)
    >>> c
    Link(1)
    
    if __________:
        (a)
        return
    next = __________
    (b)
    while __________:
        (c)
        s.rest = __________
        (d)
        s.first = __________
        (e)

    __________
    (f)
```

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

ii. (1.0 pt) Fill in blank (b)
iii. (0.5 pt) Fill in blank (c)
- s is not Link.empty
- s.rest is not Link.empty
- s is not Link.empty and s.rest is not Link.empty
- s is not Link.empty or s.rest is not Link.empty
- s.first > 0
- s.first < 0
- s.first > 1
- s.first < 1

iv. (1.0 pt) Fill in blank (d)
- 1
- s
- Link(1)
- Link(1, s)
- Link(1, s.rest)
- Link(s, Link(1))
- Link(s.first, Link(1))

v. (0.5 pt) Fill in blank (e)
- s
- s.rest
- s.rest.first
- s.first - 1
- s.first + 1
- -1 * s.first

vi. (1.0 pt) Fill in blank (f)
(c) (5.0 points) Knots

Given a linked list with only 1s, we can create knots inside the list. A knot is formed by merging multiple Links with 1s into a single Link, where the resulting Link stores the number of 1 nodes that were merged together.

For example, given a linked list of <1 1 1>, we can generate the following knotted linked lists:

<2 1 1> # Linked list with <(1 + 1), 1, 1>
<1 2 1> # Linked list with <1, (1 + 1), 1>
<1 1 2> # Linked list with <1, 1, (1 + 1)>
<2 2> # Linked list with <(1 + 1), (1 + 1)>
<3 1> # Linked list with <(1 + 1 + 1), 1>
<1 3> # Linked list with <1, (1 + 1 + 1)>
<4> # Linked list with <(1 + 1 + 1 + 1)>

Implement knot, a function that takes in a linked list s with only 1s and returns a new linked list with one or more knots. knot should return the knotted linked list that has the largest product over all of its values. If there is a tie between multiple linked lists, return any list. Assume s has at least 2 nodes.

You may use functions implemented in previous subparts (product, untangle) to solve this question.

def knot(s):
    ""
    Takes in a linked list S with only 1s and returns a new knotted linked list.
    The returned knotted linked list has the largest product over all of its values.
    If there is a tie between multiple linked lists, return any list.
    Assume S has at least 2 nodes.
    ""

    >>> a = Link(3)
    >>> untangle(a) # <1 1 1>
    >>> knot(a) # <1 1 1> --> <3>
    Link(3)
    >>> b = Link(5)
    >>> untangle(b) # <1 1 1 1 1>
    >>> knot(b) # <1 1 1 1 1> --> <2 3> or <3 2>
    Link(2, Link(3))
    >>> c = Link(10)
    >>> untangle(c) # <1 1 1 1 1 1 1 1>
    >>> knot(c) # <1 1 1 1 1 1 1 1> --> any list with two 2s and two 3s
    Link(2, Link(2, Link(3, Link(3)))))
    ""

    if _________:
        (a)
        return s
    next = s.rest
    keep_first = Link(_________, _________)
        (b)    (c)
    merge_first = knot(_________)
    (d)
    return _________([keep_first, merge_first], key=_________)
        (e)    (f)
i. (0.5 pt) Fill in blank (a)
   - s is Link.empty
   - s.rest is Link.empty
   - s is Link.empty and s.rest is Link.empty
   - s.first == 1
   - s.first != 1
   - s.first <= 1

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

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

iv. (1.0 pt) Fill in blank (d)
   - Link(next.first, next)
   - Link(next.first, next.rest)
   - Link(s.first, next)
   - Link(s.first, next.rest)
   - Link(s.first + next.first, next)
   - Link(s.first + next.first, next.rest)
   - Link(next.first + next.rest.first, next)
   - Link(next.first + next.rest.first, next.rest)

v. (0.5 pt) Fill in blank (e)
   - knot
   - sum
   - max
   - min
   - map
   - filter

vi. (1.0 pt) Fill in blank (f)
3. OOPeration

(a) (3.0 points) Patient

Leena is a hospitalist and is dissatisfied with the current system that organizes patient information at her hospitals, Kryser and Stanville, which have different admit and discharge protocols. Since some patients are admitted at multiple hospitals (somehow), the system gets confused. 

You suggest representing each patient with a specific set of attributes - name (a string), patient ID (an integer), age (an integer), and admitted, a boolean that indicates whether they are currently admitted at a hospital. 

A valid patient ID must be a seven-digit number, with the first three digits being less than 4 and the last four digits being between 0-9. 

Start by implementing the Patient class.

```python
class Patient:
    """
    >>> kim = Patient("Kim", 212764, 12)
    Invalid Patient ID entered. Try again.
    >>> kim.ID
    >>> kim.set_ID(3337656)
    Valid Patient ID
    >>> kim.ID
    3337656
    >>> tina = Patient("Tina", 2219542, 93)
    Valid Patient ID
    """
    def __init__(self, name, ID, age):
        self.name = name
        self.age = age
        self.admitted = False
        """
        (a)
        """
    def set_ID(self, ID):
        if re.match(_________, f"{ID}"):  
        (b)
            self.ID = ID
            print("Valid Patient ID")
        else:
            """
            (c)
            """
            print("Invalid Patient ID entered. Try again.")

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

ii. (1.0 pt) Fill in blank (b)
iii. (1.0 pt) Fill in blank (c)
(b) (6.0 points) Hospital

Implement the Hospital class. A Hospital instance has two instance attributes: name and patient_list, which keeps track of all the patients (Patient instances) admitted to that hospital.

class Hospital:

    >>> clinic42 = Hospital("StanKrys")
    >>> neel = Patient("Neel", 3125456, 63)
    Valid Patient ID
    >>> clinic42.admit(neel)
    >>> clinic42.admit_from_list([Patient("Ne", 3125456, 63), Patient("Sam", 1125432, 14)])
    Valid Patient ID
    Valid Patient ID
    Patient with ID 3125456 is already admitted
    >>> clinic42.patient_IDs_list()
    [3125456, 1125432]
    >>> clinic42.discharge(clinic42.find_patient(3125456))
    >>> clinic42.patient_IDs_list()
    [1125432]

    hospitals = {}
    def __init__(self, name):
        self.patient_list = []
        
        (a)
    def admit(self, patient):
        IDs = _________
        if patient.ID in IDs:
            print(f"Patient with ID {patient.ID} is already admitted")
            else:
                _________
                patient.admitted = True
        
        def admit_from_list(self, patient_list):
            for p in patient_list:
                _________
                
            (d)
        def discharge(self, patient):
            if patient.ID in self.patient_IDs_list():
                _________
                patient.admitted = False
        
        def patient_IDs_list(self):
            return _________
            
            (f)
        def find_patient(self, ID):
            for p in self.patient_list:
                if ID == p.ID:
                    return p
i. (1.0 pt) Fill in blank (a)
   - self.hospitals = self.hospitals.append(self)
   - hospitals[name] = self
   - Hospital.hospitals[name] = self
   - hospitals.extend(self)
   - self.hospitals = self.hospitals.extend(self)
   - Hospital.hospitals = {self.name: self}

ii. (1.0 pt) Fill in blank (b)
   - self.patient_list
   - patient_IDs_list(self)
   - self.patient_IDs_list
   - self.hospitals.extend(self)
   - self.patient_IDs_list(self.patient_list)
   - self.patient_IDs_list()

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

iv. (1.0 pt) Select all that apply for blank (d)
   - self.admit_from_list(p)
   - self.admit(p)
   - patient_IDs_list
   - self.patient_list.append(patient)
   - self.patient_list.extend(patient)
   - p.admit(self)

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

vi. (1.0 pt) Fill in blank (f)
(c) (7.0 points)
Implement the Kryser and Stanville classes.

Here are the protocols:
- An ID is unique to each patient so Patient("Sam", 1234511, 61) is the same as Patient("Samantha", 1234511, 60). This ensures the system doesn’t have duplicate medical charts for a patient.
- Stanville will not discharge patients who are also admitted at Kryser
- Kryser will not admit patients who are also admitted at Stanville
- There is only one Kryser Hospital and one Stanville Hospital with the names “Kryser” and “Stanville,” respectively.
- Assume these are the only two hospitals that patients can be admitted to

class Kryser(Hospital):

```python
>>> k = Kryser("Kryser")
>>> s = Stanville("Stanville")
>>> nora = Patient("Nora", 3125456, 63)
Valid Patient ID
>>> jim = Patient("Jim", 2125496, 34)
Valid Patient ID
>>> s.admit(nora)
>>> nora.admitted
True
>>> k.admit(nora) # Kryser will NOT admit Stanville patients
Error: Patient with ID 3125456 is already admitted at Stanville
>>> s.admit(nora)
>>> nora.admitted
True
>>> k.admit(nora) # Kryser will NOT admit Stanville patients
Error: Patient with ID 3125456 is already admitted at Stanville
>>> k.patient_IDs_list()
[2125496]
>>> s.admit(jim) # Stanville is okay admitting Kryser patients
>>> s.discharge(jim)
Error: Patient with ID 2125496 hasn’t been discharged from Kryser
>>> k.discharge(jim)
>>> s.discharge(jim)
>>> jim.admitted
False
```

def admit(self, patient):
    if __________ and not patient in self.patient_list:
        (a)
        print(f"Error: Patient with ID {patient.ID} is already admitted at Stanville")
    else:
        __________.admit(patient)
        (b)

def discharge(self, patient):
    __________
class Stanville(_________):
  (d)

    def admit(self, patient):
        _________
    (e)

    def discharge(self, patient):
        kryser = _________
    (f)
    if patient.admitted and patient.ID in kryser.patient_IDS_list():
        print(f"Error: Patient with ID {patient.ID} hasn’t been discharged from Kryser")
    else:
        _________
    (g)

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

(ii. (1.0 pt) Fill in blank (b)
  ○ self
  ○ Hospital
  ○ super()
  ○ self.Hospital
  ○ super

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

(iv. (1.0 pt) Fill in blank (d)
  ○ Hospital
  ○ Kryser
  ○ Hospital, Kryser
  ○ Kryser, Hospital
  ○ None
  ○ Patient
  ○ None of the answers are correct
v. (1.0 pt) Fill in blank (e)


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


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

- `self.discharge(patient)`
- `super().discharge(patient)`
- `Kryser.discharge(patient)`
- `Kryser.discharge(self, patient)`
- None
- patient
- None of the answers are correct
(d) (3.0 points) Find Patient

The system at Stanville glitched and some of the patient information (name, ID, and age) got shuffled. Leena urgently needs the ID of the patient she was going to see next. Luckily, she knows the patient’s age, that they were only admitted at Stanville, and that the product of their ID digits equals their age. Implement find_patient, which takes in the patient’s age age and returns a function, checker, to which Leena can feed in the shuffled IDs until she finds the correct ID to return the Patient object.

You may use the Hospital class and assume the Stanville hospital object exists.

def find_patient(age):
    """
    >>> s = Stanville("Stanville")
    >>> grace = s.admit(Patient("grace", 1121717, 98))
    Valid Patient ID
    >>> marie = s.admit(Patient("marie", 2121141, 16))
    Valid Patient ID
    >>> machine = find_patient(98)
    >>> machine(1211311)(1023209)(1121717) # 1 * 1 * 2 * 1 * 7 * 1 * 7 = 98
    <__main__.Patient object>
    >>> find_patient(16)(1211311)(1123113)(1111111)(2121141)
    <__main__.Patient object>
    """
    def checker(ID):
        prod = 1
        parse_digits = ID
        while parse_digits > 0:
            parse_digits, last = parse_digits // 10, parse_digits % 10
            prod *= last
        if prod == age:
            stanville = _________
            (a)
            return stanville.________
            (b)
        else:
            return _________
            (c)
        return checker

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

ii. (1.0 pt) Fill in blank (b)
iii. (1.0 pt) Select all that apply for blank (c)

- checker(age)
- find_patient
- find_patient()
- find_patient(age)
- checker
- checker(x)
- find_patient(x)
4. (8.0 points) Ratios

(a) Definition: The number of labels above tree $t$ is the number of labels along the path from this node to the root of $t$ excluding the current node.

Definition: The number of labels below tree $t$ is the number of labels within a subtree $t$ excluding the current node.

Implement `ratio`, which takes a Tree $t$ and predicate function $f$, and mutates $t$ into a Ratio Tree. A Ratio Tree replaces the value at each label with a 2-element list. Index 0 holds the number of labels above this node for which $f(label)$ returns true. Index 1 holds the number of labels below this node for which $f(label)$ returns true.

For example, consider the label 2 in the first branch of the root of the tree shown below. 2 has one label along the path to the root that is odd (1). 2 has one label within its subtrees that are odd (1). As a result, the label is replaced with the list $[1, 1]$

```python
def ratio(t, f):
    """Mutate each label of t to the ratio of the number of labels above it to the number of labels below it, represented as a 2-element list - provided that f(label) == true."""
    >>> t = Tree(15, [Tree(1, [Tree(8)]), Tree(5, [Tree(4, [Tree(6), Tree(7, [Tree(9)])])])])
    >>> pred = lambda x: x % 2 == 1  # odd?
    >>> ratio(t, pred)
    >>> t
    Tree([0, 4], [Tree([1, 0], [Tree([1, 0])]), Tree([1, 0], [Tree([1, 1], [Tree([2, 0])])])])

    def helper(t, filtered_depth):
        above = (a)
        if (b)
            (c)
        below = (d)
        for b in t.branches:
            (e)
        below += (f)
        t.label = [above, below]
    return helper(t, 0)
```
i. (1.0 pt) Fill in blank (a)

ii. (1.0 pt) Fill in blank (b)
- t.label
- t.is_leaf()
- f(t.label)
- f(t.branches)
- any([f(b.label) for b in t.branches])
- all([f(b.label) for b in t.branches])

iii. (1.0 pt) Fill in blank (c)
- return f(t.label)
- return above
- t.label = [above, 0]
- t.label = [0, 0]
- filtered_depth += 1
- above += 1

iv. (1.5 pt) Fill in blank (d)
- len([b for b in t.branches if f(b.label)])
- sum([b for b in t.branches if f(b.label)])
- sum([b.label for b in t.branches])
- sum([helper(b, filtered_depth) for b in t.branches])
- len([helper(b, filtered_depth) for b in t.branches])
- all([helper(b, filtered_depth) for b in t.branches])

v. (2.0 pt) Fill in blank (e). You cannot use and, or, if, or else.
vi. **(1.5 pt)** Fill in blank (f)

- `len([b for b in t.branches if f(b.label)])`
- `sum([b for b in t.branches if f(b.label)])`
- `sum([b.label[1] for b in t.branches])`
- `len([b.label[0] for b in t.branches if f(b.label)])`
- `sum([helper(b, filtered_depth) for b in t.branches])`
- `len([helper(b, filtered_depth) for b in t.branches])`
5. (8.0 points) Kirby Consume

(a) Consumable?

Kirby is stuck in a forest of Whispy Woods. In order to get out, he needs to consume Maxim Tomatoes from vines along the way. However, tomato vines are picky about temperature and only grow tomatoes above or below a certain threshold temperature.

In fact, each vine can be represented using these three attributes: indicator, threshold, and tomatoes. A vine grows tomatoes if one of the following is true:

- It has a positive indicator and the current temperature is strictly above the threshold.
- It has a negative indicator and the current temperature is strictly below the threshold.

You may assume that an indicator is never 0.

If the vine can grow tomatoes, Kirby will consume a certain number of tomatoes from the vine, indicated by the third attribute of a vine object.

We choose to use the following Scheme data abstraction to represent a vine:

Constructor: (make-vine ind thres tomatoes) creates an vine object with the given indicator, threshold, and tomatoes.

Selectors:

- (get-ind vine) returns the vine’s indicator
- (get-thres vine) returns the vine’s threshold
- (get-tomatoes vine) returns the vine’s number of tomatoes

Note: You may assume that the vine ADT has been properly implemented already.

Implement consumable?, which takes in an vine object, vine, and the current temperature, temp. It returns #t if the vine can be consumed from given the current temperature, and #f otherwise. (consumable? (make-vine -1 31 10) 0) evaluates to #t, as the current temperature of 0 is below the threshold of 31 and the indicator of -1 is negative.

; doctests
(define v (make-vine -1 31 10))
(expect (consumable? v 0) #t) ; -1 is negative and 0 < 31
(expect (consumable? v 32) #f) ; -1 is negative and 32 > 31

(define (consumable? vine temp)
  ( __________
    temp
    (get-thresh vine))
)

i. (2.0 pt) Fill in the blank. **Hint:** Your expression should evaluate to an operator.
(b) (6.0 points) Kirby Consume

Implement `kirby-consume`, which takes in a list of vine objects `vines`, the current temperature `temp`, and `min-tomatoes`, the minimum amount of tomatoes Kirby needs to consume from the vines in order to get out. It returns `#t` if Kirby was able to consume enough tomatoes to get out and `#f` otherwise.

Note: You may assume that `consumable?` has been implemented correctly regardless of your solution to the previous part, and you may call it in your implementation for this part. You **may leave lines blank** and can specify so by answering `Blank` for the question. Your implementation **must be tail recursive**. Any non-tail recursive solution will receive no credit.

**Hint:** First try writing a tail recursive solution without looking the multiple choice options.

```scheme
; doctests
define v1 (make-vine 1 20 2))
define v2 (make-vine 1 25 5))
define v3 (make-vine -1 30 3))
define vines (list v1 v2 v3))
(expect (kirby-consume vines 20 8) #f) ; 0 + 0 + 3 < 8
(expect (kirby-consume vines 40 7) #t) ; 2 + 5 + 0 = 7
(expect (kirby-consume vines 27 9) #t) ; 2 + 5 + 3 > 9

(define (kirby-consume vines temp min-tomatoes)
  (if (null? vines)
      __________
      (a)
      (begin
        (define v (car vines))
        __________
        (b)
        __________
        (c)
        __________
        (d)
        __________)))
      (e))

i. (1.0 pt) Fill in blank (a)
  ○ 0
  ○ #t
  ○ #f
  ○ min-tomatoes
  ○ (> min-tomatoes 0)
  ○ (<= min-tomatoes 0)
```
ii. (1.0 pt) Fill in blank (b).
   - cons
   - kirby-consume
   - and
   - or
   - temp
   - <
   - >
   - Blank

iii. (1.0 pt) Fill in blank (c).
   - (cdr vines)
   - vines
   - (car vines)
   - nil
   - temp
   - (>= min-tomatoes 0)
   - (<= min-tomatoes 0)
   - Blank

iv. (1.0 pt) Fill in blank (d).
   - 0
   - temp
   - (get-ind (car vines))
   - (get-tomatoes (car vines))
   - (kirby-consume (cdr vines) temp (- min-tomatoes (get-tomatoes v)))
   - (kirby-consume (cdr vines) temp min-tomatoes)
   - Blank

v. (2.0 pt) Fill in blank (e).
   - (+ min-tomatoes (get-tomatoes v))
   - (- min-tomatoes (get-tomatoes v))
   - (if (consumable? v temp) min-tomatoes 0))
   - (and (consumable? v temp) (kirby-consume (cdr vines) temp min-tomatoes))
   - (or (consumable? v temp) (kirby-consume (cdr vines) temp min-tomatoes))
   - (- min-tomatoes (if (consumable? v temp) (get-tomatoes v) 0))
   - (+ min-tomatoes (if (consumable? v temp) (get-tomatoes v) 0))
   - Blank
6. (9.0 points)  Matchmaking

(a) (6.0 points)  do_match_form

Modify scheme to support the new special form match. A match expression has the following form: (match <subject> <clause1> <clause2> ...)

A <clause> has the following form ((<case> <consequent>). A <subject> is any expression.

The match expression first evaluates <subject>.

Starting with the first clause, evaluates <case>. If it is scheme equal to the evaluated subject, i.e. (equal? <subject> <case>) is true, evaluate the <consequent> and return it. If they are not scheme equal, proceed to the next clause. If there are no more clauses, the return value is undefined.

```scheme
scm> (define op 'add)
scm> (match op ...
...> ('mul (* 1 2))
...> ('add (+ 1 2))
...> ('sub (- 1 2)))
3
scm> (match op ...
...> ((print 'mul) (print '2))
...> ('add (+ 1 2))
...> ('(print 'sub) (- 1 2)))
mul
3
scm> (match op ('mul (* 1 2) (* 3 4))); More than two elements in a <clause>
Traceback (most recent call last):
  0  (match op ('mul (* 1 2) (* 3 4)))
Error: too many operands in form
```

Assume that the rest of the Scheme interpreter has been implemented correctly and only do_match_form needs to be implemented. You may use the following functions defined in the Scheme interpreter project: validate_form, scheme_equalp, scheme_eval

```python
def do_match_form(expressions, env):
    """Evaluate a match form.
    >>> env = create_global_frame()
    >>> do_match_form(read_line("( c (a 1) (b (+ 1 1)) (c 3))"), env)
    3
    """
    validate_form(expressions, 1) # Ensures there's at least 1 operand
    subject = _________
    (a)
    val = scheme_eval(subject, env)
    expressions = expressions.rest
    while _________:
        (b)
        pair = expressions.first
        _________
        (c)
        match_term = _________
        (d)
        if _________(val, match_term):
            (e)
            return scheme_eval(_________, env)
        (f)
        expressions = expressions.rest
```
i. (0.5 pt) Fill in blank (a)

ii. (1.0 pt) Fill in blank (b)
- expressions is not nil
- expressions.rest is not nil
- expressions.first
- len(expressions) > 0
- len(expressions) >= 0

iii. (1.0 pt) Fill in blank (c)
- scheme_eval(subject, env)
- scheme_eval(pair, env)
- scheme_equal(expressions, val)
- scheme_equal(pair, val)
- validate_form(expressions, 2, 2)
- validate_form(pair, 2, 2)

iv. (1.5 pt) Fill in blank (d)

v. (1.0 pt) Fill in blank (e)
- scheme_eval
- validate_form
- scheme_equalp
- do_match_form
- min

vi. (1.0 pt) Fill in blank (f)
(b) (3.0 points)  Match Counter

Assume the interpreter supports the \texttt{match} special form and \texttt{do-match-form} is implemented correctly. Let us consider interpreting the following expressions:

\begin{verbatim}
scm> (define op 'add)
scm> (match op
scm>     ...> ('mul (* 1 2))
scm>     ...> ('add (+ 1 2))
scm>     ...> ('sub (- 1 2)))
\end{verbatim}

The following questions relate to the previous \texttt{match} expression. Each clause is in context of the given \texttt{match} expression.

\textbf{i. (1.0 pt)} How many times is \texttt{scheme-eval} called for just this clause of the given \texttt{match} special form:

\begin{verbatim}
('mul (* 1 2))
\end{verbatim}

\textbf{ii. (1.0 pt)} How many times is \texttt{scheme-eval} called for just this clause of the given \texttt{match} special form:

\begin{verbatim}
('add (+ 1 2))
\end{verbatim}

\textbf{iii. (1.0 pt)} How many times is \texttt{scheme-eval} called for just this clause of the given \texttt{match} special form:

\begin{verbatim}
('sub (- 1 2))
\end{verbatim}
7. (9.0 points) Hills

(a) (9.0 points) Hill

Implement the generator function `hill` which takes in a positive integer `n` and returns a generator that yields every subsequence of `n` where each digit is exactly 1 away from its adjacent digits. The order in which numbers are yielded does not matter. Assume all digits in the number are unique.

```python
def hill(n):
    """
    Accepts a positive integer N, and returns a generator that
    yields every subsequence of N where each digit is exactly 1
    away from its adjacent digits.
    >>> sorted(list(hill(354)))
    [3, 4, 5, 34, 54]
    >>> sorted(list(hill(246))) # individual digits are hills themselves
    [2, 4, 6]
    >>> sorted(list(hill(32451)))
    [1, 2, 3, 4, 5, 21, 32, 34, 45, 321, 345]
    """
    
    ---------
    (a)
    if n >= 10:
        ---------:
            (b)
            ---------
                (c)
                if _________ == 1:
                    (d)
                    ---------
                        (e)

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

ii. (1.5 pt) Fill in blank (b).

  ○ for x in hill(n - 1)
  ○ for x in hill(n // 10)
  ○ for x in range(n)
  ○ for x in range(n + 1)
  ○ while True
  ○ while n > 0
  ○ if n > 0
  ○ if n % 10
iii. (2.0 pt) Fill in blank (c).

iv. (1.5 pt) Fill in blank (d).
- $x \div \text{pow}(10, n)$
- $n \div x$
- $\text{abs}(x - n \% 10)$
- $\text{abs}(x \div 10 - n \% 10)$
- $\text{abs}(x \% 10 - n \% 10)$
- $\text{abs}(n)$
- $x \% 10 - n \div 10$

v. (2.0 pt) Fill in blank (e).
8. (10.0 points) List Methods

(a) (5.0 points) Pad

Implement pad which takes in a list s of length m, and adds el to the end of the list until there are exactly n elements. If length m is greater than n, evaluate to the same list. Example: (pad '(1 2 3) 6 4) evaluates to (1 2 3 4 4 4). (pad '(1 2 3) 0 5) evaluates to (1 2 3). You cannot use and, or, begin, cond, or if in the blanks.

(define (pad s n el)
  (cond
    ((= n 0) _________)
    (a)
    ((null? s) (cons _________ _________))
    (b) (c)
    (else (cons _________ _________))))
    (d) (e)

i. (0.5 pt) Fill in blank (a)

ii. (0.5 pt) Fill in blank (b)
○ s
○ (cdr s)
○ (car s)
○ el
○ n

iii. (1.5 pt) Fill in blank (c)

iv. (0.5 pt) Fill in blank (d)
○ s
○ (cdr s)
○ (car s)
○ el
○ n

v. (1.5 pt) Fill in blank (e)
(b) (5.0 points) Shorten

The \texttt{list-concat} procedure takes in a list \texttt{a} and a list \texttt{b}, and returns the result of appending each element from list \texttt{b} to the end of \texttt{a}. Below is the implementation for reference:

\begin{verbatim}
(define (list-concat a b)
  (if (null? a)
      b
      (cons (car a) (list-concat (cdr a) b))))
\end{verbatim}

Implement \texttt{shorten} which takes in a list \texttt{s} of length \texttt{m}, and removes elements from the end of the list until there are exactly \texttt{n} elements. If length \texttt{m} is less than \texttt{n}, return the same list. (\texttt{shorten (list 1 2 3 4 5) 3}) evaluates to \texttt{(1 2 3)}. You \textbf{must} use the \texttt{list-concat} procedure defined previously. You \textbf{cannot} use \texttt{and}, \texttt{or}, \texttt{begin}, \texttt{cond}, or \texttt{if} in the blanks.

\begin{verbatim}
(define (shorten s n)
  (define (shorten-tail s n t)
      (if _________
          (a)
          _________
          (b)
          (shorten-tail (cdr s) (- n 1) _________)))
  (shorten-tail s n nil))
\end{verbatim}

i. (0.5 pt) Fill in blank (a)
   \begin{itemize}
   \item (or (null? s) (= n 0))
   \item (and (null? s) (= n 0))
   \item (and s (= n 0))
   \item (null? s)
   \item (= n 0)
   \item (< n 0)
   \end{itemize}

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

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

iv. (1.5 pt) What is the space complexity of \texttt{shorten} with respect to \texttt{n}? Assume \texttt{n} < \texttt{m}.
   \begin{itemize}
   \item \(\theta(\log(n))\)
   \item \(\theta(1)\)
   \item \(\theta(n)\)
   \item \(\theta(n^2)\)
   \item \(\theta(2^n)\)
   \end{itemize}
v. (1.5 pt) What is the time complexity of `shorten` with respect to \( n \)? Assume \( n < m \).

- \( \theta(\log(n)) \)
- \( \theta(1) \)
- \( \theta(n) \)
- \( \theta(n^2) \)
- \( \theta(2^n) \)
9. (10.0 points) S QUBE L

(a) (10.0 points)

The (abridged version) of the table below is named ranks. It stores the best result that a competitor has achieved in competition for each event in official competitive Rubik's Cube speedsolving competitions. The three columns have the following descriptions:

- **personId**: An identifier for each individual person in the ranks table. All rows are stored as a string
- **eventId**: The corresponding event of each row. 333 refers to regular 3x3x3 Cube speedsolving, while 333oh refers to 3x3x3 Cube speedsolving with one hand. All rows are stored as a string
- **result**: The result (in centiseconds) of each row. All rows are stored as an integer

<table>
<thead>
<tr>
<th>personId</th>
<th>eventId</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018KHN28</td>
<td>222</td>
<td>101</td>
</tr>
<tr>
<td>2017TSVE02</td>
<td>222</td>
<td>112</td>
</tr>
<tr>
<td>2022VISH01</td>
<td>222</td>
<td>113</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2018WIDJ01</td>
<td>333</td>
<td>669</td>
</tr>
<tr>
<td>2013BALI01</td>
<td>333</td>
<td>670</td>
</tr>
<tr>
<td>2016OLZI01</td>
<td>333</td>
<td>670</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2016SONG03</td>
<td>333oh</td>
<td>1276</td>
</tr>
<tr>
<td>2015WILS05</td>
<td>333oh</td>
<td>1276</td>
</tr>
<tr>
<td>2016SMUL01</td>
<td>333oh</td>
<td>1277</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Write a query that outputs all competitors with a faster result in One-Handed solving (333oh) than regular 3x3x3 speedsolving (333). Your query should also output the following:

- The personId corresponding to each result
- The fastest 3x3x3 Cube One Handed result in seconds, with a column name of 333oh_best
- The fastest 3x3x3 Cube result in seconds, with a column name of 333_best
- The output should be ordered in ascending order by the fastest One Handed result.

Your output will look similar to the following:

<table>
<thead>
<tr>
<th>personId</th>
<th>333oh_best</th>
<th>333_best</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017TUNG13</td>
<td>9.28</td>
<td>9.35</td>
</tr>
<tr>
<td>2021TAKA01</td>
<td>14.38</td>
<td>18.07</td>
</tr>
<tr>
<td>2015HEER02</td>
<td>14.98</td>
<td>15.38</td>
</tr>
<tr>
<td>2013ZHAN31</td>
<td>16.9</td>
<td>19.03</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

You may assume that every row in the ranks table will have a valid result.

You may not use LIKE in your solution.

**Hint**: To convert centiseconds to seconds, you need to divide the column by 100.
i. (1.0 pt) Fill in blank (a).

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

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

iv. (0.5 pt) Fill in blank (d).

v. (0.5 pt) Fill in blank (e).

vi. (5.0 pt) Fill in blank (f).

vii. (1.0 pt) Fill in blank (g).
10. (3.0 points) Special Topics

(a) (1.0 pt) ComP(ile)ython

In CPython, what is the order of intermediate results when going from source code to output?

- source->AST->output
- source->bytecode->output
- source->machine code->bytecode->output
- source->AST->tokens->bytecode->output
- source->tokens->AST->bytecode->output
- source->tokens->AST->machine code->bytecode->output

(b) (1.0 pt) BLAST OFF

Assume your query sequence is TACTGGATA which encodes for the insulin gene.

What is displayed when you run the BLAST algorithm (like we did on the NCBI website) on your query sequence and a database of 10 other insulin gene target sequences?

- Sequence with the highest genetic similarity to the query sequence using local alignment
- Sequence with the highest genetic similarity to the query sequence using global alignment
- Sequence with the highest genetic similarity to the query sequence using local or global alignment
- Highest genetic similarity of the query sequence with the target sequences using local alignment
- Highest genetic similarity of the query sequence with the target sequences using global alignment
- Genetic similarities of the query sequence with all target sequences using local alignment
- Genetic similarities of the query sequence with all target sequences using global alignment
- Top 3 highest genetic similarities of the query sequence with target sequences using global alignment

(c) (1.0 pt) Pickle Tree

Given the Tree class on the Study Guide provided, select the option to fill in blank (a) the __reduce__ function that would be used by calling pickle.dumps(tree_obj):

```python
def __reduce__(self):
    return __________
    (a)
```

- (self.__class__, (self.label, self.branches))
- (__class__, (label, branches))
- [self.__class__, [self.label, self.branches]]
- [__class__, [label, branches]]
- self
- str(self)
- repr(self)
11. (0.0 points) A+ Questions

(a) (0.0 points) A+ Levels

This A+ question is not worth any points. It can only affect your course grade if you have a high A and might receive an A+. Finish the rest of the exam first!

**Definition:** The level of a list refers to how deeply nested it is. Example: (1) has a level of 1. ((2)) has a level of 1. The inner list of (2) has a level of 2.

Implement `levels` that takes in a list `s`. It returns a new levelled list such that at each depth level `L` of the list there are exactly `L` elements. If there are more than `L` elements. Keep only the first `L` elements. If there are fewer than `L` elements. Add the depth level `L` as an element until there are exactly `L` elements.

You can use `shorten` and `pad` in your implementation. You cannot use `and`, `or`, `begin`, `cond`, or `if` in the blanks. Examples:

```
scm> (levels '(((8 1) (4 9 8) ()))) ; add 3 at depth 3. Remove empty list at depth 2. (((8 1 3) (4 9 8)))
scm> (levels '(())) ; add 3s at depth 3. Add 2s at depth 2. (((3 3 3) 2))
```

```
(define (levels s)
  (define (helper s level)
    (cond
     ((integer? s) _________) (a)
     ((null? s) _________) (b)
     (else 
      (begin 
       (define first _________)  (c)
       (define rest _________)  (d)
       (________))) ) (e)
    (helper s 1))
```
i. (0.0 pt) Fill in blank (a)

ii. (0.0 pt) Fill in blank (b)

iii. (0.0 pt) Fill in blank (c)

iv. (0.0 pt) Fill in blank (d)

v. (0.0 pt) Fill in blank (e)
(b) (0.0 points)  A+ SQL

This A+ question is not worth any points. It can only affect your course grade if you have a high A and might receive an A+. Finish the rest of the exam first!

You are given the following tables:

results

<table>
<thead>
<tr>
<th>eventId</th>
<th>result</th>
<th>personName</th>
<th>personId</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>333oh</td>
<td>1272</td>
<td>Andrew Bae</td>
<td>2014BAEA01</td>
</tr>
<tr>
<td>333oh</td>
<td>1318</td>
<td>Justin Mallari</td>
<td>2010MALL01</td>
</tr>
<tr>
<td>333oh</td>
<td>1592</td>
<td>Eric Zhao</td>
<td>2010ZHAO19</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>444</td>
<td>2917</td>
<td>Sebastian Weyer</td>
<td>2010WEYE02</td>
</tr>
<tr>
<td>444</td>
<td>2995</td>
<td>Sebastian Weyer</td>
<td>2010WEYE02</td>
</tr>
<tr>
<td>444</td>
<td>2973</td>
<td>Feliks Zemdegs</td>
<td>2009ZEMD01</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>skewb</td>
<td>1205</td>
<td>Riley Woo</td>
<td>2007WOOR01</td>
</tr>
<tr>
<td>skewb</td>
<td>1327</td>
<td>Tiffany Chien</td>
<td>2012CHIE01</td>
</tr>
<tr>
<td>skewb</td>
<td>1347</td>
<td>Jeong-Soo Park</td>
<td>2014PARK03</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

persons

<table>
<thead>
<tr>
<th>name</th>
<th>country</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Barrett Maun</td>
<td>USA</td>
<td>2022MAUN01</td>
</tr>
<tr>
<td>Ben Yin</td>
<td>USA</td>
<td>2022YINB01</td>
</tr>
<tr>
<td>Bowen Wang</td>
<td>USA</td>
<td>2022WANG56</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

results is a table that has 4 columns:

- **eventId**: The corresponding event of each row. Similarly to the previous SQL question, 333oh refers to 3x3x3 Cube speed solving with one hand. All rows are stored as a string
- **result**: The result (in centiseconds) of each row. All rows are stored as an integer
- **personName**: The name for each individual person in the results table. All rows are stored as a string
- **personId**: A unique identifier for each individual person in the results table. All rows are stored as a string

persons is a table that has 3 columns:

- **name**: The name for each individual person in the persons table. All rows are stored as a string
- **country**: The country that a person represents in competition. All results are stored as a string
- **id**: A unique identifier for each individual person in the persons table. All rows are stored as a string

Assume that personId in results will always have a match for id in persons

Write a query that outputs the number of times that a person has achieved a result strictly under 10 seconds (1000 centiseconds) for the 333oh eventId. Your query should also output the following:

- The personName corresponding to each result.
- The personId corresponding to each result.
- The country that each person represents.
- The fastest 3x3x3 Cube One-Handed result that the corresponding competitor achieved in seconds, with a column name of best_result
- The number of times that the corresponding competitor achieved a result under 10 seconds.
- The output should be sorted in descending order by the number of times someone has achieved a result under 10 seconds.
- There should be 8 rows outputted.

Your output will look similar to the following:

<table>
<thead>
<tr>
<th>personName</th>
<th>personId</th>
<th>country</th>
<th>best_result</th>
<th>num_times_sub_10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Park</td>
<td>2012PARK03</td>
<td>USA</td>
<td>8.76</td>
<td>63</td>
</tr>
<tr>
<td>Patrick Ponce</td>
<td>2012PONC02</td>
<td>USA</td>
<td>8.65</td>
<td>18</td>
</tr>
<tr>
<td>Luke Garrett</td>
<td>2017GARR05</td>
<td>USA</td>
<td>9.18</td>
<td>12</td>
</tr>
<tr>
<td>Dwayne Ramos</td>
<td>2019RAMO05</td>
<td>New Zealand</td>
<td>8.88</td>
<td>9</td>
</tr>
<tr>
<td>Sean Patrick Villanueva</td>
<td>2017VILL41</td>
<td>Philippines</td>
<td>9.13</td>
<td>8</td>
</tr>
<tr>
<td>Juliette Sébastien</td>
<td>2014SEBA01</td>
<td>France</td>
<td>9.23</td>
<td>7</td>
</tr>
<tr>
<td>Magnus Lensch</td>
<td>2019LENS01</td>
<td>Germany</td>
<td>9.33</td>
<td>6</td>
</tr>
<tr>
<td>Zhouheng Sun</td>
<td>2008SUNZ01</td>
<td>China</td>
<td>9.81</td>
<td>4</td>
</tr>
</tbody>
</table>

SELECT ________________
   (a)
FROM _______, _______
   (b) (c)
WHERE ________________
   (d)
GROUP BY _______
   (e)
ORDER BY _______
   (f)
LIMIT _______;
   (g)

i. (0.0 pt) Fill in blank (a).

ii. (0.0 pt) Fill in blank (b).

iii. (0.0 pt) Fill in blank (c).
iv. (0.0 pt) Fill in blank (d).

v. (0.0 pt) Fill in blank (e).

vi. (0.0 pt) Fill in blank (f).

vii. (0.0 pt) Fill in blank (g).
12. (0.0 points) Fun Question

(a) (0.0 pt) WWYD with the Link\text{.rest} of your summer. (draw \text{'(a picture)}). Alternatively, feel free to draw a picture about 61A!
No more questions.