INSTRUCTIONS

- The exam is worth 75 points, across 7 questions.
- You should have 6 sheets in your exam booklet.
- You have 2 hours to complete the exam. **Do NOT open the exam until you are instructed to do so!**
- The exam is closed book, closed notes, closed computer, closed calculator, except two hand-written 8.5” × 11” crib sheets of your own creation and the official CS 88 Final study guide.
- Mark your answers on the exam itself. We will not grade answers written on scratch paper.

<table>
<thead>
<tr>
<th>Full Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student ID Number</td>
<td></td>
</tr>
<tr>
<td>Official Berkeley Email (@berkeley.edu)</td>
<td></td>
</tr>
<tr>
<td>TA</td>
<td></td>
</tr>
<tr>
<td>Name of the person to your left</td>
<td></td>
</tr>
<tr>
<td>Name of the person to your right</td>
<td></td>
</tr>
<tr>
<td><em>By my signature, I certify that all the work on this exam is my own, and I will not discuss it with anyone until exam session is over.</em> (please sign)</td>
<td></td>
</tr>
</tbody>
</table>

POLICIES & CLARIFICATIONS

- If you need to use the restroom, bring your phone and exam to the front of the room.
- You may use built-in Python functions that do not require import, such as `min`, `max`, `pow`, `len`, and `abs`.
- You **may not** use example functions defined on your study guide unless a problem clearly states you can.
- For fill-in-the-blank coding problems, we will only grade work written in the provided blanks. You may only write one Python statement per blank line, and it must be indented to the level that the blank is indented.
- Unless otherwise specified, you are allowed to reference functions defined in previous parts of the same question.
1. (12 points) What Would Python Display?

For each of the expressions in the table below, 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”. The first two rows have been provided as examples.

The interactive interpreter displays the value of a successfully evaluated expression, unless it is None.

Assume that you have first started python3 and executed the statements on the left.

they = lambda x: return len(x())

def cant(y, x):
    return lambda: [y(x + 1)
        for i in range(x) if x % 2 == 0]

def stop(naruto_running):
    if naruto_running > 0:
        print('SASUKEEE')
    return 2019

of = ['high', 'school']
of = of + [of[0] + of[1]]
us = len(of)
def all(in_this, together):
    if 'highschool' in in_this:
        return 2
    if together % 7 == 3:
        return 3
    else:
        print('musical')

class Clown:
    fles = 1
    def __init__(self, se, lf):
        self.se = lf
        se.lf = self
        self.fles = Clown.fles
        Clown.fles += self.fles

stop = Clown(Clown, Clown)
no = Clown(stop, Clown)
why = Clown(no, stop)

<table>
<thead>
<tr>
<th>Expression</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>they</td>
<td>Function</td>
</tr>
<tr>
<td>[ (y,y) for y in [x for x in range(3)] ][0]</td>
<td></td>
</tr>
</tbody>
</table>
| sleep = lambda zzz: zzz * 8
cant(sleep, len('well'))() | |
| Clown.fles | |
| why.se == why.lf | |
| assert isinstance(stop, Clown) | |
2. (8 points) Save the Environment!

Fill in the environment diagram that results from executing the code on the right until the entire program is finished, an error occurs, or all frames are filled. A complete answer will:

- Add all missing names and parent annotations to all local frames.
- Add all missing values created or referenced during execution.
- Show the return value for each local frame.
- Hint: Integer division means the floor of \( x/y \)

```python
x = [100, lambda budget: x[0]]
def how (vacation):
    return vacation // 10 # integer division
def food (money, budget):
    money = x[0]
    work = x[1]
def how(food):
    return how is your(food)
def your(fav):
    while fav == 100:
        x = lambda fav: work(money) - 2
        fav = x(fav)
x.append(5)
return how
vacationPrice, discount = how(x[0]), 50
is_food = food(vacationPrice, discount)(100)
```

![Environment Diagram](image-url)
3. **(10 points) SQL: CS88Rising**

We have compiled information on popular artists that are a part of the popular music group, 88rising. Knowledge of the group is not required to answer the following questions. Given these following two tables, complete the following SQL queries so that they return the requested data. (“Hard coding” answers to return only the exact data will not receive credit.)

**Table: Artists**

<table>
<thead>
<tr>
<th>stage_name</th>
<th>real_name</th>
<th>place_of_birth</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joji</td>
<td>George Miller</td>
<td>USA</td>
<td>27</td>
</tr>
<tr>
<td>NIKI</td>
<td>Nicole Zefanya</td>
<td>Indonesia</td>
<td>20</td>
</tr>
<tr>
<td>Rich Brian</td>
<td>Brian Soewarno</td>
<td>Indonesia</td>
<td>20</td>
</tr>
<tr>
<td>Higher Bros.</td>
<td>更高兄弟</td>
<td>China</td>
<td>25</td>
</tr>
<tr>
<td>Oski Bear</td>
<td>Oski</td>
<td>USA</td>
<td>79</td>
</tr>
</tbody>
</table>

**Table: Songs**

<table>
<thead>
<tr>
<th>title</th>
<th>artist</th>
<th>length (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>History</td>
<td>Rich Brian</td>
<td>307</td>
</tr>
<tr>
<td>Head in the Clouds</td>
<td>Joji</td>
<td>178</td>
</tr>
<tr>
<td>Indigo</td>
<td>NIKI</td>
<td>173</td>
</tr>
<tr>
<td>Hold Me Down</td>
<td>Higher Bros.</td>
<td>223</td>
</tr>
<tr>
<td>Sanctuary</td>
<td>Joji</td>
<td>180</td>
</tr>
</tbody>
</table>

A. Write a SQL query that retrieves the `stage_name` of all artists who are not from the USA.

____________________________________________________________________________________________

B. Write a SQL query that lists the `real_name` and `place_of_birth` of all artists in descending order based on their age.

____________________________________________________________________________________________

C. Write a SQL query that lists the `title` and `real_name` of a song’s artist in descending order based on the `length` of the respective song.

____________________________________________________________________________________________

____________________________________________________________________________________________

D. Write a SQL query that gets all the `title`s of all unique pairs of songs where both their respective artists are younger than 27.
4. (6 points) Lists of Lists of Lists of Lists...

Write a remove function that takes in a linked list and a number and returns a new linked list with that number removed in the new linked list. If the element does not exist in the linked list, return a copy of the original linked list.

```python
>>> lnk_lst = Link(1, Link(2, Link(3, Link(4, Link(5))))))
>>> remove(lnk_lst, 2)
Link(1, Link(3, Link(4, Link(5))))
>>> lnk_lst
Link(1, Link(2, Link(3, Link(4, Link(5))))))

def remove(linked_lst, num):
    return
```

What is the runtime of this function with respect to the length of the list? (Bubble one)

- [ ] Constant
- [ ] Logarithmic
- [ ] Linear
- [ ] Quadratic
- [ ] Exponential

```python
class Link:
    """Reference of the Link class."
    ""
    empty = ()

    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest

    def __len__(self):
        return 1 + len(self.rest)

    def __repr__(self):
        if self.rest:
            rest_str = ', ' + repr(self.rest)
        else:
            rest_str = ''
        return 'Link({0}{1})'.format(self.first, rest_str)
```
(6 points) Get Your Numbers in a Row

Complete the `isStrictlyIncreasing` function and its helper function below to return `True` if a number has strictly increasing digits from left to right and `False` otherwise. You may assume a single digit number is specified to be strictly increasing and that the number is always nonnegative.

Use the following lines of code, which are currently out of order as a basis for your solution. You may need to fill in the blanks.

```python
def isStrictlyIncreasing(num):
    
    >>> isStrictlyIncreasing(8)
    True
    >>> isStrictlyIncreasing(21)
    False
    >>> isStrictlyIncreasing(133)
    False
    >>> isStrictlyIncreasing(569)
    True
    
def helper(n, prev):
    _________________________________________________________________
    _________________________________________________________________
    _________________________________________________________________
    _________________________________________________________________
    _________________________________________________________________
    _________________________________________________________________
    return helper(_________________________, ______________________)
```

```python
def isStrictlyIncreasing(num):
    
    return True
    return helper(____________, ______________)
    return False
    if n == ___________
    else:
        if prev ____  ________:
```
6. (15 points) One Hungry Artist
You are a famous artist and for your newest exhibit you decided to tape a banana to a wall. The 
banana starts out unripe, but when it ripens someone comes and eats your banana, and then you can 
replace banana back with an unripe banana once it is eaten. The banana in the artpiece can only be 
“UNRIPE”, “RIPE”, or “EATEN”.

def create_art(state):
    """Create the art""
    return { 'banana?': [state] }

Use the following set of doctests to understand the behavior of the functions on the next page.

>>> art = create_art("UNRIPE")
>>> is_unripe(art)
True
>>> set_banana("RIPE", art)
>>> is_ripe(art)
True
>>> is_unripe(art)
False
>>> is_eaten(art)
False
>>> eat_banana(art) # We eat our ripe banana
>>> is_ripe(art)
False
>>> is_eaten(art)
True
>>> ripen_banana(art) # Does nothing
>>> exhibit = generator(art)
>>> next(exhibit)
"There is no banana."
>>> next(exhibit)
"There is no banana."
>>> replace_banana(art)
>>> next(exhibit)
"Eww, unripe banana."
>>> ripen_banana(art)
>>> next(exhibit)
"The banana looks so delicious."
>>> set_banana("Broken Exhibit", art) # set to a not expected value
>>> next(exhibit) # cause a StopIteration Exception
StopIteration: Unknown banana state
Fill out the rest of the functions to get the corresponding doctest to have the right output.

```python
def set_banana(banana, art):
    'Update the art'

    ____________________________________________________________________________________________

def is_ripe(art):
    "Return whether the art's internal state is 'RIPE'"

    ____________________________________________________________________________________________

def is_unripe(art):
    "Return whether the art's internal state is 'UNRIPE'"

    ____________________________________________________________________________________________

def is_eaten(art):
    "Return whether the art's internal state is 'EATEN'"

    ____________________________________________________________________________________________

For the next set of functions, you must not violate the abstraction barrier! Any abstraction violations will have points deducted.

```python
def ripen_banana(art):
    'Only ripen banana if it is unripe'

    ____________________________________________________________________________________________

```

def replace_banana(art):
    'Only replace banana if it is eaten'

    ____________________________________________________________________________________________

```

def eat_banana(art):
    'Only eat banana if it is ripe'

    ____________________________________________________________________________________________

```
def generate_art(art):
    ""
    Yield the corresponding string based on the state of the banana.
    Raise an Exception if the art is in an unknown state.
    (Remember, you may not need all lines).
    ""

    while True:
        ...
7. **(10 points) We've Inherited Some Baking Skills**

It's dessert time again! Fill in the `Chocolate` and `Cake` classes based on the doctests. Both these classes should inherit from our base `Dessert` class. Be sure to take advantage of inheritance whenever possible. You may not need all lines provided.

```python
>>> cake = Cake('cake', 2, 2, 'chocolate')
>>> cake.eat()
Oh no! There is no more cake
>>> cake.bake()
Adding chocolate
Baking...
>>> cake.eat()
Yum, I ate 2 pieces of cake
>>> cake.eat()
Oh no! There is no more cake
>>> chocolate = Chocolate('chocolate', 5, 3)
>>> chocolate.eat()
Yum, I ate 3 pieces of milk chocolate
>>> chocolate.eat()
Oh no! There is no more milk chocolate
>>> chocolate.happiness
False

class Dessert:
    def __init__(self, name, portions_remaining, portion_size):
        self.name = name
        self.happiness = True
        self.portions_remaining = portions_remaining
        self.portion_size = portion_size

    def eat(self):
        if self.portions_remaining >= self.portion_size:
            self.portions_remaining -= self.portion_size
            print("Yum, I ate " + str(self.portion_size) + " pieces of " + self.name)
        else:
            print("Oh no! There is no more " + self.name)
```
class ____________________________:
    def __init__(self, name, portions_remaining, portion_size):
        super().______________________________

    def eat(self):
        ________________________________
        ________________________________
        Dessert.______________________________

Class ____________________________:
    def __init__(self, name, portions_remaining, portion_size, ____________):
        Dessert.______________________________

        self.portions_to_be_made = ________________________________

    def bake(self):
        ________________________________
        ________________________________
        ________________________________