This 90-minute closed-book, closed-notes exam has 6 questions worth a total of roughly 77 points (some point-total adjustment may occur during grading).
You may separate the pages while working on the exam; we have a stapler available.

It is a violation of the Academic Integrity Code to look at any exam other than your own, to look at any reference material besides the 1 page reference provided, or to otherwise give or receive unauthorized help.
We also ask that you not discuss this exam with students who are scheduled to take a later makeup.
Academic Integrity is expected of all students of Cornell University at all times, whether in the presence or absence of members of the faculty. Understanding this, I declare I shall not give, use or receive unauthorized aid in this examination.

Signature: ___________________________________________ Date __________

First Name: __________________________________________

Last Name: __________________________________________

Cornell NetID, all caps: ____________________________________
1. [8 points] **Strings.** Implement the following function.

```python
def peel(markers, text):
    """Returns a new string where the `markers` have been removed from the
    beginning and end of `text`.

    Examples:
    peel( "()", "(abc)" ) --> "abc"
    peel( "()", "(1(+))" ) --> "1(+)1"
    peel( "<>()", "<>(.<>)" ) --> ">.<"
    peel( "ab", "ab" ) --> ""

    Preconditions:
    markers: string of even length (0 is allowed)
    text: any-length string that starts w/ 1st half of `markers`, ends w/ 2nd half.
    """
    # REMINDER: in a slice expression like s[n:m], n and m must be ints, not floats
```
def swap2(a_list, j, k):
    """Modifies a_list by swapping the two elements of a_list starting
    at index j with the 2 entries of a_list starting at index k.
    """
    Examples:
    swap2([100, 101, 102, 103, 104, 105, 106, 107, 108, 109], 1, 6)
    changes a_list to
    [100, 106, 107, 103, 104, 105, 101, 102, 108, 109]
    """--------"""

    swap2([100, 101, 102, 103, 104, 105, 106, 107, 108, 109], 0, 4)
    changes a_list to
    [104, 105, 102, 103, 100, 101, 106, 107, 108, 109]
    """--------"""

    swap2(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'], 0, 4)
    changes a_list to
    ['e', 'f', 'c', 'd', 'a', 'b', 'g', 'h', 'i', 'j']
    """--------"""

    Preconditions:
    j and k are valid indices (positive, < len(a_list))
    j + 2 <= k (the elements you're swapping don't overlap in a_list)
    k + 2 <= len(a_list) """
3. Some truths are self evident. Some are learned in CS 1110.

   (a) [2 points] True or False? The drawing below accurately depicts the value of variable x in Global Memory after the code below is executed in Python:

   ```python
def double_x(input):
    return input * 2
x = 5
x = double_x(x)
```

   Circle One:

   True   False

   (b) [2 points] True or False? The drawing below accurately depicts the value of variable x in Global Memory after the code below is executed in Python:

   ```python
def double_x(input):
    x = input * 2
x = 5
double_x(x)
```

   Circle One:

   True   False

   (c) [2 points] True or False? The drawing below accurately depicts the value of variable x in Global Memory after the code below is executed in Python:

   ```python
def double_x(input):
    x = input * 2
    print(str(x))
x = 5
double_x(x)
```

   Circle One:

   True   False

   (d) [2 points] True or False? The drawing below accurately depicts the value of variable x in Global Memory after the code below is executed in Python:

   ```python
def double_x(input):
    x = input * 2
    print(str(x))
x = 5
x = double_x(x)
```

   Circle One:

   True   False
4. [24 points] **Time for dinner!** Place is an object with 3 attributes: spoon, fork, and knife. A call of the form `Place(s,f,k)` creates a new Place object with attribute spoon set to `s`, fork set to `f`, and knife set to `k`. Assume that class Place is accessible within the given code. Simulate running all 27 lines of code and draw the memory diagram as seen in class and Assignment 2.

```python
def soup(p):
    p.spoon = p.spoon + 1
    drawer.spoon = drawer.spoon - 1

def salad(p):
    p.fork = p.fork + 1
    drawer.fork = drawer.fork - 1
    p2.knife = p2.knife + 2
    drawer.knife = drawer.knife - 1

def dinner(p, with_soup, with_salad):
    if with_soup:
        soup(p)
    if with_salad:
        salad(p)

def dessert(p, name):
    if name == "ice cream":
        n_spoons = 2
    else:
        n_spoons = 0
    p.fork = p.fork + 1
    p.spoon = p.spoon + n_spoons
    return n_spoons

p1 = Place(1, 2, 0)
p2 = Place(1, 2, 0)
drawer = Place(6, 4, 8)
dinner(p1, False, True)
n_spoons = dessert(p2, "ice cream")
drawer.spoon = drawer.spoon - n_spoons
```

---

**Global Space**

**Heap**

**Call Stack**
5. [8 points] Testing, Testing, 1, 2, 3, Testing!

Consider the following function specification, which you might use if you want to distribute the cost of dinner amongst you and your friends.

```python
def batch_withdraw(balance_list, withdraw_amount):
    """balance_list is a list of floats representing the balances of multiple bank accounts

    Pre-condition:
    withdraw_amount is a float with value >= 0.

    Return a new list of the same length as balance_list, where every value is the corresponding value in balance_list minus withdraw_amount. If any value in balance_list is less than withdraw_amount (i.e., there is not enough in the account to withdraw), return the empty list. """
```

Here is an example of one set of sample inputs and an expected output:

<table>
<thead>
<tr>
<th>Test Case</th>
<th>balance_list</th>
<th>withdraw_amount</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[20.0, 30.0, 40.0, 50.0]</td>
<td>10.0</td>
<td>[10.0, 20.0, 30.0, 40.0]</td>
</tr>
</tbody>
</table>

Provide two more conceptually distinct test cases, using the same format. Include a short statement (1-2 sentences) explaining what situation each of your test cases represents.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>balance_list</th>
<th>withdraw_amount</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Test Case 2 covers the following situation:

<table>
<thead>
<tr>
<th>Test Case</th>
<th>balance_list</th>
<th>withdraw_amount</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Test Case 3 covers the following situation:
6. **The eyes have it.** Assume objects of class `Point` have two attributes: `x` and `y`; both are `int`s. Assume objects of new class `Face` have three `Point` attributes: `left_eye`, and `right_eye`, and `nose`. `Face` attributes should have the following relationships to be considered **proportionate**:

- `left_eye` and `right_eye` have the same `y` attribute values (they are the same height)
- `left_eye` and `right_eye` are centered across the `y`-axis (`left_eye`'s `x` attribute is negative and `right_eye`'s `x` attribute is positive)
- `nose` always sits on the `y`-axis (`x`=0)
- `nose` is always lower than the eyes by the distance that the eyes are from the `y`-axis. 
  Example: if the eyes are 2 units from the `y`-axis, the nose will be 2 units below the eyes.

![An example of a proportionate face.](image)

(a) [6 points] Implement the following function.

```python
def set_face(f, right_x, right_y):
    """Given ints right_x and right_y (which are the desired values for the x and y coordinates of the right eye of Face f), sets the left_eye, right_eye and nose attributes of Face f, so that Face f is proportionate.

    Precondition: right_x and right_y are non-negative ints."
    # Reminder: to negate the variable n in Python, you simply write -n.
```
(b) [9 points] Implement the following function.

def is_proportionate(f):
    """Return True if the locations of the eyes and nose of Face f make the face `proportionate`, based on the definition at the beginning of this question. If any of the x,y attributes of the elements of Face f are not in proportion, return False."
    """

(c) [6 points] Implement the following function.

def eyes_wider(first, second):
    """Return True if the eyes of Face `first` are wider apart than the eyes of Face `second`. Otherwise return False. Also return False if either face is not proportionate."
    """