It is a violation of the Academic Integrity Code to look at any exam other than your own, to look at any other reference material, 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 __________

This 90-minute exam has 6 questions worth a total of 66 points. When permitted to begin, scan the whole test before starting. Budget your time wisely. Use the back of the pages if you need more space. You may tear the pages apart; we have a stapler at the front of the room.

You may not use explicit for-loops or recursion on this exam. Beyond that, you may use any Python feature that you have learned about in class (if-statements, map, lists, and so on), except: please use `str` instead of backquotes (handwritten backquotes often look like single quotes).

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>66</td>
<td></td>
</tr>
</tbody>
</table>

The Important First Question:

1. [2 points] When allowed to begin, write your last name, first name, and Cornell NetID at the top of each page, and circle your lab time on the top of this page.
2. [16 points] Match the shaded parts of the following Python program to the names below. In your answer, each letter should occur exactly once.

```python
# Format the date as a string.
date_str = month_names[dt.month] + ' ' + str(dt.day) + ', ' + str(dt.year)

# Adjust the time for 12-hour clock if required.
hour = dt.hour
if twelve_hour:
    time_suffix = (' AM' if dt.hour < 12 else ' PM')
    hour = (12 if hour == 0 else (hour if hour <= 12 else hour - 12))
else:
    time_suffix = ''

# Format the time part of the string.
time_str = ':'.join(map(str, [hour, dt.minute, dt.second]))

# Assemble the result from the values computed so far
return date_str + ' ' + time_str + time_suffix
```

A. `month_names` = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
                      'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

B. `def date_time_str(dt, twelve_hour):
```
C. `***Return the date and time in the object <dt>, in the format
   MMM DD, YYYY HH:MM:SS
   Example:
   Mar 7, 2013 19:30:00
   if the boolean <twelve_hour> is True, the time is given in 12-hour
   format with AM or PM appended. Example:
   Mar 7, 2013 7:30:00 PM
   ***
```
D. `date_str = month_names[dt.month] + ' ' + str(dt.day) + ', ' + str(dt.year)`

E. `# Format the date as a string.`
F. `# Adjust the time for 12-hour clock if required`
```
G. `hour = dt.hour`
H. `if twelve_hour:
```
I. `time_suffix = (' AM' if dt.hour < 12 else ' PM')`
J. `hour = (12 if hour == 0 else (hour if hour <= 12 else hour - 12))`
K. `else:
```
L. `time_suffix = ''`
M. `# Format the time part of the string`
N. `time_str = ':'.join(map(str, [hour, dt.minute, dt.second]))`
```
O. `# Assemble the result from the values computed so far
return date_str + ' ' + time_str + time_suffix`
```
```
```
O. `print date_time_str(get_current_time(), True)`
3. [16 points] Two students were assigned to diagram the execution of the following code. You are their grader; please circle all errors and write in anything that is missing. You may wish to do this question by first drawing the relevant frames and objects yourself.

```python
def f(x, y):
    c = 3*x
    y[0] = b + c + y[1]
    c = 0

def g(a, b):
    f(2, a)
    return b + a[0]
```

```python
b = 4
c = 5
x = g([b, c], 7)
```

```
3. [16 points] Two students were assigned to diagram the execution of the following code. You are their grader; please circle all errors and write in anything that is missing. You may wish to do this question by first drawing the relevant frames and objects yourself.

def f(x, y):
    c = 3*x
    y[0] = b + c + y[1]
    c = 0

def g(a, b):
    f(2, a)
    return b + a[0]
```

```python
b = 4
c = 5
x = g([b, c], 7)
```
4. [4 points] Here, we consider a simplified version of extracting information from a web page. Assume that variable \( \text{x} \) stores a string of the form

\[
\text{<a href="string1">string2</a>}
\]

where both \( \text{string1} \) and \( \text{string2} \) are strings that do not contain double quotes or angle brackets. The only space in the format shown above is after the first a, although \( \text{string1} \) and \( \text{string2} \) may themselves contain spaces. Example: if \( \text{x} \) were the string \( \text{'<a href="this">that</a>'} \), then \( \text{string1} \) would be 'this' and \( \text{string2} \) would be 'that'. Write a sequence of one or more statements that result in variable \( \text{s2} \) holding the string \( \text{string2} \).

For reference:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{s.find(s1)} )</td>
<td>Returns: index of the first character of the FIRST occurrence of ( \text{s1} ) in ( \text{s} ), or (-1) if ( \text{s1} ) does not occur in ( \text{s} ).</td>
</tr>
<tr>
<td>( \text{s.find(s1, i)} )</td>
<td>Returns: index of the first character of the FIRST occurrence of ( \text{s1} ) in ( \text{s} ) at or after position ( i ), or (-1) if ( \text{s1} ) does not occur in ( \text{s}[i:] ). If ( i ) is omitted, searches the whole string.</td>
</tr>
<tr>
<td>( \text{s.index(s1)} )</td>
<td>Like \text{find}, but raises an error if ( \text{s1} ) is not found.</td>
</tr>
<tr>
<td>( \text{s.index(s1, i)} )</td>
<td>Like \text{find}, but raises an error if ( \text{s1} ) is not found.</td>
</tr>
<tr>
<td>( \text{s.rfind(s1)} )</td>
<td>Returns: index of the first character of the LAST occurrence of ( \text{s1} ) in ( \text{s} ), or (-1) if ( \text{s1} ) does not occur in ( \text{s} ).</td>
</tr>
<tr>
<td>( \text{s.rindex(s1)} )</td>
<td>Like \text{rfind}, but raises an error if ( \text{s1} ) is not found.</td>
</tr>
</tbody>
</table>
5. This question involves code for suggesting new NetIDs.

Assume file last.py defines a type of object called LastUsed. These have two attributes:

- **prefix**: non-empty string of lowercase letters
- **suffix**: positive int

and can be created by calls like this: `last.LastUsed('djs', 98)` (if `last` has been imported).

File last.py also implements the function `ind(lulist, p)` with the following spec:

```python
def ind(lulist, p):
    """Returns: index in lulist of LastUsed object with prefix p (-1 if no such object)
    Preconds: lulist is a (possibly empty) list of LastUsed objects with distinct
    prefixes. p is a non-empty string of lowercase letters."""
```

(a) [8 points] Draw all objects and variables created by the following sequence of commands.

(Don’t draw any frames.)

```python
import last
temp = [last.LastUsed('ljl', 2), last.LastUsed('srm', 2)]
has_srm = last.ind(temp, 'srm')
```

(b) [12 points] On the next page(s), complete file nets.py by following the helpful directions given in curly braces. Each such direction can require multiple lines to implement. For reference, here are some functions and the like you can use:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x in lt</code></td>
<td>Returns: True if x is in list lt, False otherwise.</td>
</tr>
<tr>
<td><code>lt.append(x)</code></td>
<td>Append object x to the end of list lt.</td>
</tr>
<tr>
<td><code>lt.pop(i)</code></td>
<td>Returns: item at position i in list lt, removing it from lt. If i is omitted, returns and removes the last item.</td>
</tr>
<tr>
<td><code>lt.sort()</code></td>
<td>Sort the items of lt, in place (the list is altered).</td>
</tr>
</tbody>
</table>
The file nets.py continues on next page...
If all_last contains a LastUsed object with prefix equal to inits, then add one to that object’s suffix and assign to variable suf this new suffix value. Otherwise, add to all_last a new LastUsed object with prefix inits and suffix 1, and assign to variable suf this new suffix value.

{Return the NetID corresponding to inits and suf}
6. [8 points] Complete the body of testing procedure `testnew` for function `newid` from the previous problem. You may make at most five calls to `newid`. Our grading will focus on the completeness of your test cases: they should cover the space of possible arguments with which `newid` could be called. To save time on this exam, do not directly check whether the argument list has been correctly modified; only directly check whether `newid`'s return value is correct.

```python
import last
import nets
import cunittest2

def testnew():
    """Test the newid fn in nets""
```

For reference, here are some functions in `cunittest2`:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>assert.equals(expected, received)</code></td>
<td>Raise an AssertionError if <code>expected</code> and <code>received</code> differ.</td>
</tr>
<tr>
<td><code>assert.true(received)</code></td>
<td>Raise an AssertionError if <code>received</code> is False.</td>
</tr>
</tbody>
</table>

1Yes, for this exam we're doing the testing after the implementation. Tsk, tsk.