Last Name: Ution First Name: Sol Cornell NetID: SU1110

Circle your lab: Tu 12:20 Tu 1:25 Tu 2:30 Tu 3:35 W 12:20 W 1:25 W 2:30 W 3:35

CS 1110 Prelim 1 March 7th, 2013 SOLUTIONS

1. 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.

   Every time a student doesn't do this, somewhere, a kitten weeps.
   More seriously, we did have an exam come apart during grading, so it is actually important to write your name on each page. Also, remember that if we need to figure out your lab section at the end of the grading session (roughly 2am this time around), our chances of putting it the wrong pile, and thus you not being able to find it when you get to lab, grow high.

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.

   I Assignment statement
   Q Name of a function being called
   C Name of a function being defined
   J/R Boolean expression
   D Parameter
   H/R Argument
   K String literal
   L Integer literal
   B List
   F List indexing
   G/P Function call expression
   P Method call expression
   E Docstring
   O Comment
   M Conditional expression
   A Name of global variable being created
   N Name of local variable being created
   F/H Reference to an attribute of an object

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

def date_time_str(dt, twelve_hour):
    """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
    """
    # 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

print date_time_str(get_current_time(), True)
```

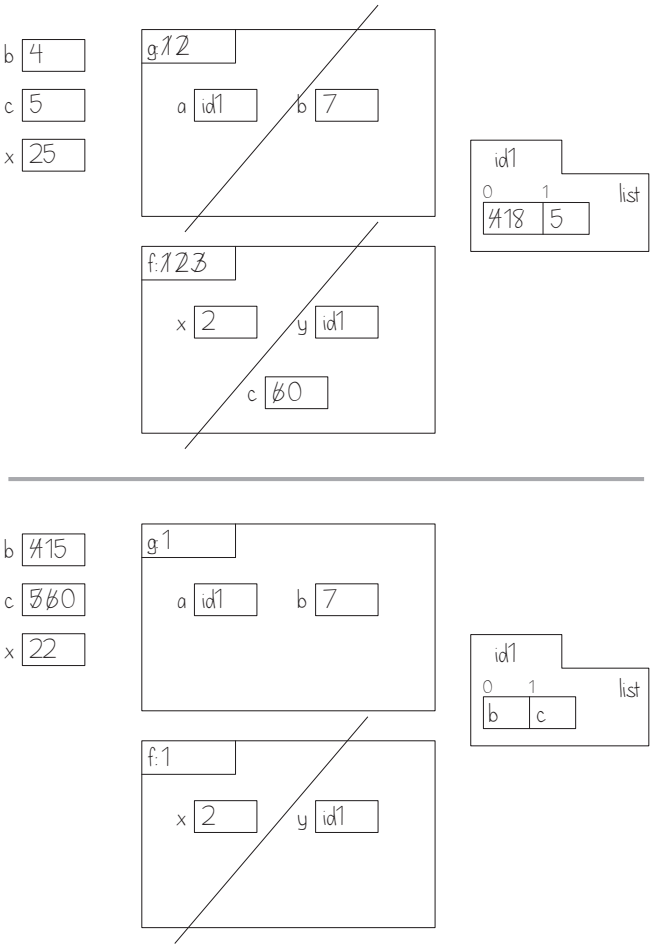
3. [16 points] Two students were assigned to diagram the execution of the following code. You are their grader; please circle all errors and write in anything that is missing. You may wish to do this question by first drawing the relevant frames and objects yourself.

In the first purported solution, the "25" should be a "22" (so the second solution got this right!) and the "18" should be 15.

```
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]

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 `x` stores a string of the form

   `<a href="`string1`">`string2`</a>`

   where both string1 and 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 string1 and string2 may themselves contain spaces. Example: if `x` were the string `'<a href=" this "> that</a>'`, then string1 would be `' this '` and string2 would be `' that'`.

   Write a sequence of one or more statements that result in variable `s2` holding the string string2.

```
# There are several possible solutions; here's one.

i1 = x.index('>')
i2 = x.rindex('<')
s2 = x[i1+1:i2]
```

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).

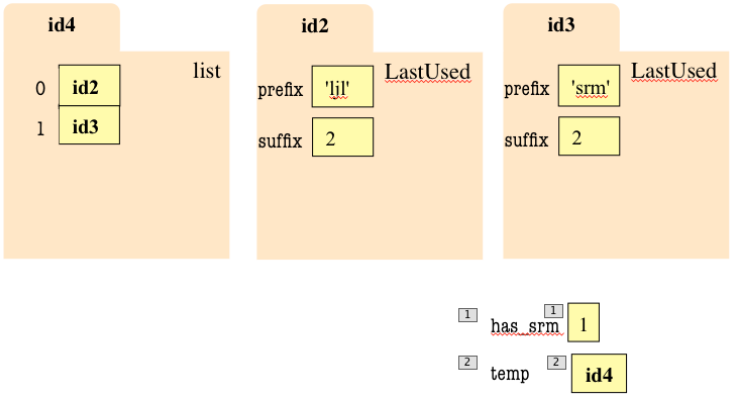
   File `last.py` also implements the function `ind(lulist, p)` with the following spec:

```
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.)

```
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:

`x in lt`	Returns: `True` if `x` is in list `lt`, `False` otherwise.
`lt.append(x)`	Append object `x` to the end of list `lt`.
`lt.pop(i)`	Returns: item at position `i` in list `lt`, removing it from `lt`. If `i` is omitted, returns and removes the last item.
`lt.sort()`	Sort the items of `lt`, in place (the list is altered).

`# nets.py` {Omit other authoring info.}

```
import last
def newid(fname, mname, lname, all_last):
    """Returns: NetID for new Cornellian named fname mname lname.  For
       people with the same initials, gives out sequentially numbered
       NetIDs starting with the number 1.

       The new NetID is a string of this person's initials (first
       initial coming first) and the next available numerical suffix,
       according to all_last.

       The list all_last keeps track of which NetIDs have been used; it
       contains a LastUsed object for each set of initials that has
       been used in a NetID, with the highest number that has been given
       out so far.  It is modified to account for the new NetID
       returned by this function.
       {Don't worry, we explain how to do this in the remarks below.}

       For instance, if all_last started out empty, and then the NetIDs
       abc1, foo1, and abc2 are generated, all_last should contain
       two LastUsed objects: one with prefix 'abc' and suffix 2, and one
       with prefix 'foo' and suffix 1.

       Preconditions: all arguments are strings containing only
           lowercase letters.  The lengths of fname and lname are at
           least 1.  The list all_last contains LastUsed objects
           indicating which NetIDs have been used."""
```

```
    # Many solutions were possible.
    # A common error was to try something like if inits in all_last. The problem is
    # that all_last is a list of LastUsed objects, not strings, and inits is a string.

    if mname == '':
        inits = fname[0] + lname[0]
    else:
        inits = fname[0] + mname[0] + lname[0]
    i = last.ind(all_last, inits) # inits is new iff i is -1

    if i != -1:
        all_last[i].suffix = all_last[i].suffix + 1
        suf = all_last[i].suffix
    else:
        all_last.append(last.LastUsed(inits,1))
        suf = 1

    return inits + str(suf)
```

6. [8 points] Complete the body of testing procedure `testnew` for function `newid` from the previous problem.[1] 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.

```
import last
import nets
import cunittest2

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

Many solutions were possible.

```
    lt = []
    # test if correct when list is empty
    cunittest2.assert_equals('srm1', nets.newid('sa', 'ra', 'max', lt))
    # test old initials
    cunittest2.assert_equals('srm2', nets.newid('sa', 'ra', 'max', lt))
    # test new initials
    cunittest2.assert_equals('ll1', nets.newid('la', '', 'lee', lt))
    # test an old set of initials, and also a LastUsed not in position 0
    cunittest2.assert_equals('ll2', nets.newid('laci', '', 'lee', lt))
    #test having a middle initial, same first and last initial
    cunittest2.assert_equals('ljl1', nets.newid('li', 'joon', 'lee', lt))
    print "All tests for newid passed"
```

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