Last Name: _____ First: _____ Netid: _____ Section _____

CS 1110 Prelim 1 October 13th, 2016

This 90-minute exam has 6 questions worth a total of 100 points. 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.

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.

You will be expected to write Python code on this exam. We recommend that you draw vertical lines to make your indentation clear, as follows:

```
def foo():
    if something:
        do something
        do more things
    do something last
```

You should not use recursion on this exam. Beyond that, you may use any Python feature that you have learned about in class (if-statements, try-except, lists, for-loops and so on), **unless directed otherwise**.

Question	Points	Score
1	2	
2	20	
3	20	
4	16	
5	22	
6	20	
Total:	100	

The Important First Question:

1. [2 points] Write your last name, first name, netid, and *lab section* at the top of each page.

Last Name: _____ First: _____ Netid: _____ Section _____

Reference Sheet

Throughout this exam you will be asked questions about strings and lists. You are expected to understand how slicing works. In addition, the following functions and methods may be useful.

Math Functions

Function	Description
`sqrt(x)`	**Returns**: The square root of `x` as a `float`.
`ceil(x)`	**Returns**: The smallest integer value greater than or equal to `x` as a `float`.
`floor(x)`	**Returns**: The largest integer value less than or equal to `x` as a `float`.

All of these functions are in the module `math`. This module must be imported to access them.

String Functions and Methods

Function or Method	Description
`len(s)`	**Returns**: number of characters in `s`; it can be 0.
`a in s`	**Returns**: True if the substring `a` is in `s`; False otherwise.
`s.find(s1)`	**Returns**: index of the first character of the FIRST occurrence of `s1` in `s` (-1 if `s1` does not occur in s).
`s.find(s1,n)`	**Returns**: index of the first character of the first occurrence of `s1` in `s` STARTING at position n. (-1 if `s1` does not occur in s from this position).
`s.count(s1)`	**Returns**: number of (non-overlapping) occurrences of substring `s1` in `s`.
`s.isalpha()`	**Returns**: True if `s` is *not empty* and its elements are all letters; it returns False otherwise.
`s.isdigit()`	**Returns**: True if `s` is *not empty* and its elements are all numbers; it returns False otherwise.
`s.strip()`	**Returns**: A *copy* of `s` with all spaces at either the beginning or end removed from `s`.

List Functions and Methods

Function or Method	Description
`len(x)`	**Returns**: number of elements in list `x`; it can be 0.
`y in x`	**Returns**: True if `y` is in list `x`; False otherwise.
`x.index(y)`	**Returns**: index of the FIRST occurrence of `y` in `x` (an error occurs if `y` does not occur in `x`).
`x.count(y)`	**Returns**: number of times `y` appears in the list `x`.
`x.append(y)`	Adds `y` to the end of list `x`.
`x.insert(i,y)`	Inserts `y` at position `i` in list `x`, shifting later elements to the right.
`x.remove(y)`	Removes the first item from the list whose value is `y`. (an error occurs if `y` does not occur in `x`).

The last three list methods are all procedures. They return the value `None`.

2. [20 points total] **Short Answer Questions**.

   (a) [6 points] What are the four kinds of variables introduced in class? Define each of them.

   (b) [3 points] What is a *fruitful function*? What is a *procedure*? Your answer should explain how they differ and how they are the same.

   (c) [3 points] Which of the following two expressions produces an error (none, one, or both)?

   ```
   False and (5 / 0)          (5 / 0) or True
   ```

   Explain why this is the case.

   (d) [4 points] What is the definition of a *type cast*? What is the difference between a *widening* cast and a *narrowing* cast? Give an example of each.

(e) [4 points] What is the difference between the following two commands?

```
>>> import math
>>> from math import *
```

Your answer should describe the affects of each command on global space.

3. [20 points total] **Testing and Debugging**.

(a) [8 points] Unlike many other languages, lists in Python can contain a mixture of types. For example, the following is legal in Python

```
s = [1, True, 'Hello', 3.0]
```

This list contains an `int`, a `bool`, a `string` and a `float`.

For reasons we will see later, we often prefer that the elements of our lists all have the same type. That is why we might have a function like the following:

```
def extract_int(seq):
    Return: a list containing only the ints in seq.

    The function returns a new list that contains only those elements of
    seq that are ints, in the same order they occur in seq.

    Example: extract_int([1, True, 'A', 2]) is [1,2]

    Precondition: seq is a list
```

Do not implement this function. Instead, write down a list of at least **four test cases** that you would use to test out this function. By a test case, we just mean an input and an expected output; you do not need to write an `assert_equals` statement. For each test case *explain why it is significantly different from the others.*

(b) [12 points] In class, we demonstrated the function `anglicize`, which turns an int in its English equivalent. Listed below is an alternate implementation for the number 100 to 999, called `ang100to999`. This function has several helpers to aid it. These functions are mostly correct. However there are at least three bugs in this code. These bugs are across all functions and are not limited to a single function.

To help find the bugs, we have added several print statements throughout the code. The result of running the code with these print statements shown on the next page. Using this information as a guide, identify and fix the three bugs on the next page. You should explain your fixes.

Hint: Do not "fix" the lines marked NOT A BUG. It is okay to break up a list this way. In addition, we recommend that you start with the test on the next page, and do not try to read the code without using the tests.

Code:

```
1  def ang100to999(n):
2      """Returns: English equiv of n.
3
4      Precond: n an int in 100..999"""
5      pref = ang1to19(n/100) + ' hundred'
6      print 'Pref: '+pref          # WATCH
7
8      tens = n % 100
9      print 'Tens: '+str(tens)     # WATCH
10     if tens > 0 and tens < 20:
11         print 'In if-statement'  # TRACE
12         suff = ' '+ang1to19(tens)
13         print 'Suff: '+suff      # TRACE
14     elif tens > 20:
15         print 'In elif-statement' # TRACE
16         suff = ' '+ang20to99(tens)
17         print 'Suff: '+suff      # TRACE
18     return pref + suff
19
20
21 def ang20to99(n):
22     """Returns: English equiv of n.
23
24     Precond: n an int in 20..99"""
25     print 'ang20to99'            # TRACE
26     pref = tens(n/10)
27     print 'Pref: '+pref          # WATCH
28     suff = ' '+ang1to19(n % 10)
29     print 'Suff: '+suff          # WATCH
30     return pref+suff
31
32
33 def tens(n):
34     """Returns: tens-word for n
35
36     Precond: n an int in 2..9"""
37     print 'tens'                 # TRACE
38     print 'n: '+str(n)           # WATCH
39     # Assignment is NOT A BUG
40     words = ['twenty','thirty','forty',
41              'fifty','sixty','seventy',
42              'eighty','ninety']
43     result = words[n-1]
44     print 'Result: '+result      # WATCH
45     return result
46
47
48 def ang1to19(n):
49     """Returns: English equiv of n.
50
51     Precond: n an int in 1..19"""
52     print 'ang1to19'             # TRACE
53     print 'n: '+str(n)           # WATCH
54     # Assignment is NOT A BUG
55     words = ['one','two','three','four',
56          'five','six','seven','eight',
57          'nine','ten','eleven',
58          'twelve','thirteen','fourteen',
59          'fifteen','sixteen','seventeen',
60          'eighteen','nineteen']
61     result = words[n-1]
62     print 'Result: '+result      # WATCH
63     return result
```

Tests:

```
>>> ang100to999(134)
ang1to19
n: 1
Result: one
Pref: one hundred
Tens: 34
In elif-statement
ang20to99
tens
n: 3
Result: forty
Pref: forty
ang1to19
n: 4
Result: four
Suff: four
Suff: forty four
'one hundred forty four'
```

```
>>> ang100to999(120)
ang1to19
n: 1
Result: one
Pref: one hundred
Tens: 20
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "angl.py", line 18, in ang100to999
    return pref + suff
UnboundLocalError: local variable 'suff'
referenced before assignment
```

```
>>> ang100to999(130)
ang1to19
n: 1
Result: one
Pref: one hundred
Tens: 30
In elif-statement
ang20to99
tens
n: 3
Result: forty
Pref: forty
ang1to19
n: 0
Result: nineteen
Suff: nineteen
Suff: forty nineteen
'one hundred forty nineteen'
```

First Bug:

Second Bug:

Third Bug:

4. [16 points] **String Slicing**. When announcing the release date for a new product, companies have to pay attention to the official date format. In the US our date format is `mm/dd/yy` (e.g. month, day, year). However, Europe often uses the format `dd/mm/yy` (e.g. day, month, year). For example, a product released on 3/6/16 in the US might be released 9/3/16 in Europe, which is just 3 days later.

   As shown in the example above, sometimes the month and day may only be a single digit. However, we typically pad these out to two digits with a leading zero. With this in mind, complete the function below using what you know about strings.

```
def europeanize(date):
    Returns: European version of this date (type is a string).

    Days and months are padded (if necessary) to become two digits each.

    Examples:
       europeanize('3/6/12') is '06/03/12'
       europeanize('01/29/11') is '29/01/11'

    Precondition: date a string representing a US date.
```

Last Name: _____ First: _____ Netid: _____ Section _____

5. [22 points] **Call Frames.**

Consider the following function definitions.

```
1  def cutout(seq):
2      pos = lookup(seq)
3      a = seq[:pos]
4      b = seq[pos+1:]
5      return a+b

6  def lookup(it):
7      return it[0]
8
```

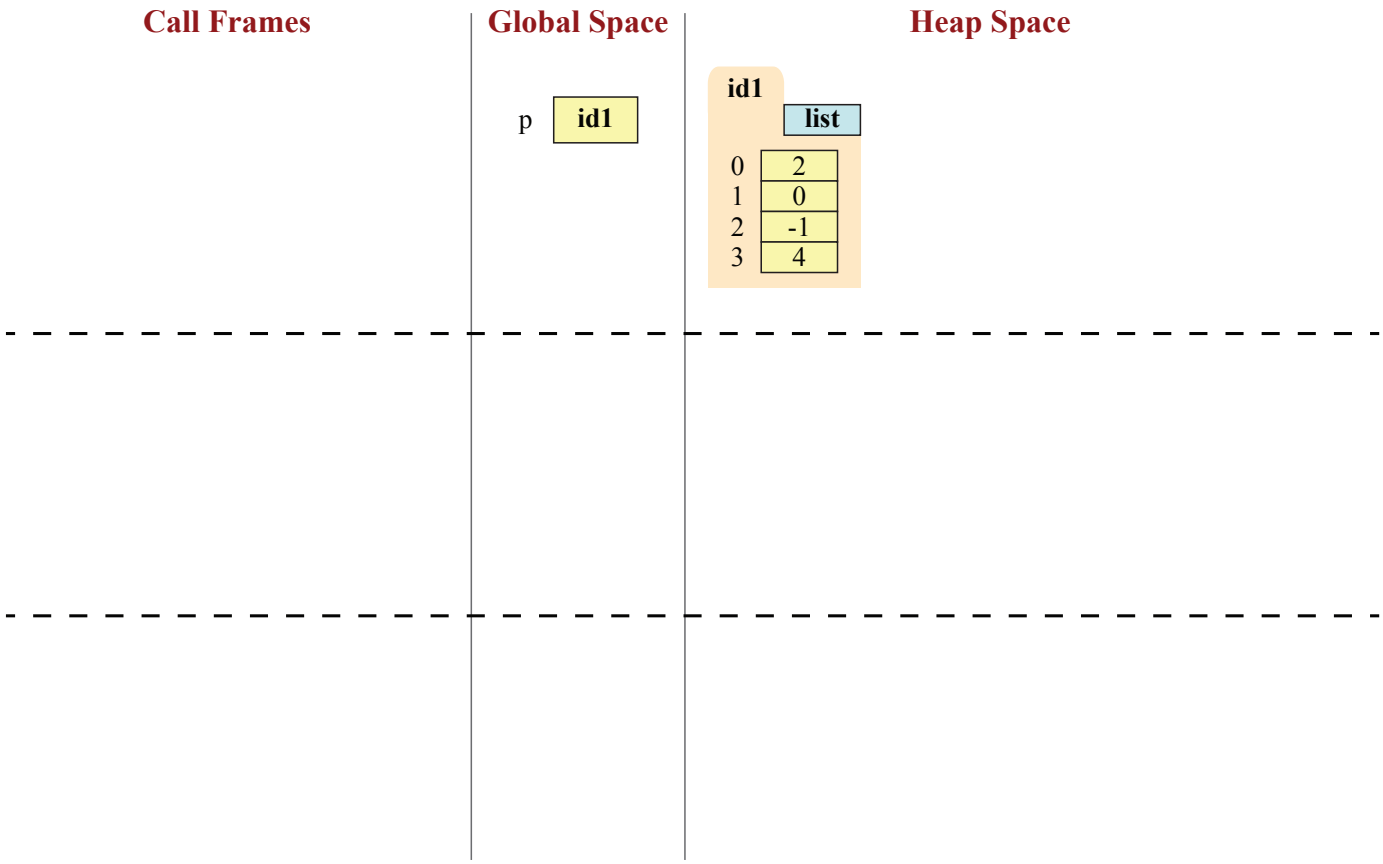
Assume that `p = [2,0,-1,4]` is a global variable that stores a reference to a list in heap space, as shown below. On this page and the next, diagram the evolution of the call

```
p = cutout(p)
```

Diagram the state of the *entire call stack* for the function `cutout` when it starts, for each line executed, and when the frame is erased. If any other functions are called, you should do this for them as well (at the appropriate time). This will require a total of **eight** diagrams, not including the first one below.

In addition, you should draw the state of global space and heap space at each step. You can ignore the folders for the function definitions. Only draw folders for lists or objects. You are also allowed to write "unchanged" if no changes were made to either global or heap space.

Hint: Pay close attention to the line numbers. They are different than those in the assignment.

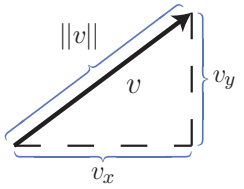


Last Name: _____ First: _____ Netid: _____ Section _____

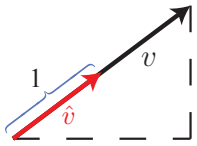
Call Frames **Global Space** **Heap Space**

6. [20 points total] **Objects and Functions**.

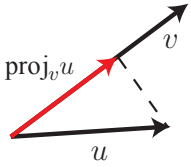
Two dimensional vectors have a lot of applications in computer graphics. A vector v is defined by its components (v_x, v_y), as shown in the picture below.



(a) Vector Norm



(b) Unit Vector



(c) Vector Projection

There are several important functions on vectors.

Dot Product and Norm. The dot-product takes two vectors and combines them into single value, as given by the following formula.

$$v \cdot u = v_x u_x + v_y u_y$$

The length (or norm) of vector is the square root of the dot-product with itself, as per the following formula.

$$||v|| = \sqrt{v \cdot v} = \sqrt{v_x v_x + v_y v_y}$$

The norm is nonzero so long as either v_x and v_y are non-zero.

Unit Vector. For any given vector v, the unit vector $\hat{v}$ is the vector with the same direction as v, but with a length of 1. It is defined by the following formula.

$$\hat{v} = \left(\frac{v_x}{||v||}, \frac{v_y}{||v||} \right)$$

Note that this only defined if v is not the zero vector.

Vector Projection. For any two vector v and u, the projection of u onto v is written $\text{proj}_v u$. This is a vector with same direction of v, but whose length forms a right angle with the vector u, as shown in picture (c) above. We compute the projection with the following formula:

$$\text{proj}_v u = \left(\frac{v_x(u \cdot v)}{v \cdot v}, \frac{v_y(u \cdot v)}{v \cdot v} \right)$$

Again, this assumes that v is not zero.

In this problem you will be working with the class `Vec2` which provides two dimension vectors in Python. This is a class with exactly two attributes – `x` and `y` – specifying the vector components. The only invariant for these attributes is that they must be floats. You create a new `Vec2` with the constructor function `Vec2(x,y)` which assigns the attributes in that order.

One the next page, you will implement functions computing the unit vector and vector projection. You can assume that the module providing `Vec2` is already imported, and you do not need to use the module as a prefix when calling the constructor (for example, you can just call `Vec2(1,2)`. Implement these functions exactly as specified.

(a) [10 points] **Unit Vector**

```
def normalize(v):
    Changes the vector v into a unit vector in the same direction.

    This function is a procedure.  It does not return a new vector.  It
    modifies the x and y attributes of the parameter v.  If v is a zero
    vector, the function leaves it unchanged.

    Preconditions: v is a Vec2 object.
```

(b) [10 points] **Vector Projection**

```
def project(u,v):
    Returns: the projection of u onto v.

    This function returns a new vector and does not modify either u or v.
    If v is the zero vector, then it returns a new zero vector.

    Preconditions: u and v are Vec2 objects.
```