

Last Name: _____ First: _____ Netid: _____

CS 1110 Prelim 1 October 17th, 2024

This 90-minute exam has 6 questions worth a total of 100 points. Read over 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 loops or recursion on this exam. Beyond that, you may use any Python feature that you have learned in class (if-statements, try-except, lists), **unless directed otherwise**.

Question	Points	Score
1	2	
2	15	
3	18	
4	25	
5	18	
6	22	
Total:	100	

The Important First Question:

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

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.

String Functions and Methods

Expression or Method	Description
<code>len(s)</code>	Returns: number of characters in <code>s</code> ; it can be 0.
<code>a in s</code>	Returns: True if the substring <code>a</code> is in <code>s</code> ; False otherwise.
<code>s.count(s1)</code>	Returns: the number of times <code>s1</code> occurs in <code>s</code>
<code>s.find(s1)</code>	Returns: index of the first character of the FIRST occurrence of <code>s1</code> in <code>s</code> (-1 if <code>s1</code> does not occur in <code>s</code>).
<code>s.find(s1,n)</code>	Returns: index of the first character of the first occurrence of <code>s1</code> in <code>s</code> STARTING at position <code>n</code> . (-1 if <code>s1</code> does not occur in <code>s</code> from this position).
<code>s.rfind(s1)</code>	Returns: index of the first character of the LAST occurrence of <code>s1</code> in <code>s</code> (-1 if <code>s1</code> does not occur in <code>s</code>).
<code>s.isalpha()</code>	Returns: True if <code>s</code> is <i>not empty</i> and its elements are all letters; it returns False otherwise.
<code>s.isdigit()</code>	Returns: True if <code>s</code> is <i>not empty</i> and its elements are all numbers; it returns False otherwise.
<code>s.isalnum()</code>	Returns: True if <code>s</code> is <i>not empty</i> and its elements are all letters or numbers; it returns False otherwise.
<code>s.islower()</code>	Returns: True if <code>s</code> is <i>has at least one letter</i> and all letters are lower case; it returns False otherwise (e.g. <code>'a123'</code> is True but <code>'123'</code> is False).
<code>s.isupper()</code>	Returns: True if <code>s</code> is <i>has at least one letter</i> and all letters are upper case; it returns False otherwise (e.g. <code>'A123'</code> is True but <code>'123'</code> is False).
<code>s.lower()</code>	Returns: A copy of <code>s</code> with all letters lower case.
<code>s.upper()</code>	Returns: A copy of <code>s</code> with all letters upper case.

List Functions and Methods

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

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

Last Name: _____ First: _____ Netid: _____

2. [15 points total] **Short Answer Questions.**

(a) [5 points] Name the four categories of variables we have seen in class. Describe each one.

(b) [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.

(c) [6 points] Consider the following code:

```
1 def first(n):
2     print('Start first')
3     try:
4         second(n)
5         print('In first try')
6     except:
7         print('In first except')
8     print('Done first')
9
10 def second(n):
11     print('Start second')
12     try:
13         assert n <= 0, 'is not <= 0'
14         print('In second try')
15     except:
16         print('In second except')
17     assert n >= 0, 'not >= 0'
18     print('Done second')
```

What is printed out when we call the following functions?

i. `first(-1)`

ii. `first(1)`

Last Name: _____ First: _____ Netid: _____

3. [18 points] **String Slicing.**

Implement the function below according the specification. Remember to refer to the reference guide on the second page.

```
def isrepeated(s):  
    """Returns whether a '-' dash divides s in to equal substrings.  
  
    If '-' is not in s, or appears more than two times, this function returns False.  
    Otherwise, we use the dash to break s into (2 or 3) substrings. We return True  
    only if these substrings are equal.  
  
    Examples:  
        isrepeated('a-a') returns True  
        isrepeated('ab-ab-ab') returns True  
        isrepeated('a-A') returns False  
        isrepeated('a-aa') returns False  
        isrepeated('a') returns False  
        isrepeated('a-a-a-a') returns False  
  
    Precondition: s is a string"""
```

4. [25 points total] **Testing and Debugging.**

- (a) [10 points] Consider the following function header and specification:

```
def extract_int(seq):
    """Returns a list containing only the ints in seq, in the same order as seq

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

    Precondition: seq is a list"""
```

Do not implement this function. Instead, write down a list of at least **six 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) [9 points] In the Roman numeral system, the symbols 'I', 'V', 'X', 'L', and 'C' stand for the the numbers 1, 5, 10, 50, and 100, respectfully. A symbol **after** another of equal or greater value adds its value (e.g. 'VI' is 6, 'XI' is 11, and 'XV' is 15). A symbol **before** one of greater value subtracts its value (e.g. 'IV' is 4, 'IX' is 9, and 'XC' is 90). If you are not familiar with the Roman numeral system, see the chart below for relevant examples.

The function `romanize` takes a integer 1..99 and converts it into a Roman numeral. There are **at least three bugs** in the code on the next page. To help find the bugs, we have added several print statements throughout the code, and show the results on the page after that. Using this information as a guide, identify and fix the three bugs on the answer page. Your fixes may include more than one line of code. You should explain your fixes.

Arabic	Roman	Arabic	Roman	Arabic	Roman	Arabic	Roman
7	'VII'	36	'XXXVI'	59	'LIX'	80	'LXXX'
14	'XIV'	40	'XL'	60	'LX'	83	'LXXXIII'
20	'XX'	44	'XLIV'	64	'LXIV'	90	'XC'
23	'XXIII'	50	'L'	70	'LXX'	92	'XCII'
30	'XXX'	52	'LII'	77	'LXXVII'	99	'XCIX'

Note: Write your answers on the next page

```

1 def romanize(n):
2     """Returns the Roman numeral for n
3
4     Example: romanize(74) returns 'LXXIV'
5
6     Precond: 0 < n < 100 is an int"""
7     tens = ''
8     ones = ''
9     if n >= 50:
10         print('More than 50')      # TRACE
11         tens = numeralL(n//10)
12     elif n >= 10:
13         print('More than 10')      # TRACE
14         tens = numeralX(n//10)
15     print('tens = '+repr(tens))    # WATCH
16
17     ones = romanize1to9(n % 10)
18     print('ones = '+repr(ones))    # WATCH
19     return tens+ones
20
21
22 def romanize1to9(n):
23     """Returns the Roman numeral for n
24
25     Example: romanize1to9(5) returns 'V'
26
27     Precond: 0 < n < 10 is an int"""
28     # Combined TRACE and WATCH
29     print('romanize1to9: n = '+repr(n))
30     if n < 5:
31         print('Less than 5')      # TRACE
32         return romanize1to4(n)
33     elif n < 9:
34         print('Between 5 and 8')  # TRACE
35         return 'V'+romanize1to4(n-5)
36     else:
37         print('Equal to 9')      # TRACE
38         return 'IX'
39
40
41 def romanize1to4(n):
42     """Returns the Roman numeral for n
43
44     Example: romanize1to4(3) returns 'III'
45
46     Precond: 0 < n < 5 is an int"""
47     # Combined TRACE and WATCH
48     print('romanize1to4: n = '+repr(n))
49     values = ['I', 'II', 'III', 'IV']
50     choiceI = values[n-1]
51
52     # WATCH
53     print('choiceI = '+repr(choiceI))
54     return choiceI
55
56 def numeralL(n):
57     """Returns Roman numeral for tens value
58
59     The value n is the tens DIGIT of n
60     Example: numeralL(6) returns 'LX'
61
62     Precond: 5 <= n < 10 is an int"""
63     # Combined TRACE and WATCH
64     print('numeralL: n = '+repr(n))
65
66     if n < 9:
67         print('Less than 90')      # TRACE
68         return 'L'+numeralX(n-5)
69     else:
70         print('Equals to 90')      # TRACE
71         return 'XC'
72
73
74 def numeralX(n):
75     """Returns Roman numeral for tens value
76
77     The value n is the tens DIGIT of n
78     Example: numeralX(3) returns 'XXX'
79
80     When n is 0, it returns the empty
81     string (to be compatible w/ numeralL)
82
83     Precond: 0 <= n < 5 is an int"""
84     # Combined TRACE and WATCH
85     print('numeralX: n = '+repr(n))
86
87     values = ['', 'X', 'XX', 'XXX', 'XL']
88
89     choiceX = values[n]
90     # WATCH
91     print('choiceX = '+repr(choiceX))
92     return choiceX
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110

```

Last Name: _____ First: _____ Netid: _____

Hint: Some bugs cannot be fixed with just one line. You might need to add a conditional.

Tests:

```
>>> romanize(14) # Expected: 'XIV'
More than 10
numeralX: n = 1
choiceX = 'X'
tens = ''
romanize1to9: n = 4
Less than 5
romanize1to4: n = 4
choiceI = 'IV'
ones = 'IV'
'IV'
```

First Bug:

```
>>> romanize(75) # Expected: 'LXXV'
More than 50
numeralL: n = 7
Less than 90
numeralX: n = 2
choiceX = 'XX'
tens = 'LXX'
romanize1to9: n = 5
Between 5 and 8
romanize1to4: n = 0
choiceI = 'IV'
ones = 'VIV'
'LXXVIV'
```

Second Bug:

```
>>> romanize(60) # Expected: 'LX'
More than 50
numeralL: n = 6
Less than 90
numeralX: n = 1
choiceX = 'X'
tens = 'LX'
romanize1to9: n = 0
Less than 5
romanize1to4: n = 0
choiceI = 'IV'
ones = 'IV'
'LXIV'
```

Third Bug:

(c) [6 points] **Do not implement the function specified below.**

Instead, use assert statements to enforce the precondition. Furthermore, each the assert statement should produce one of the three error messages shown below

```
>>> before_space('abc')
AssertionError: 'abc' has no spaces.
>>> before_space(' abc')
AssertionError: ' abc' has an illegal space.
>>> before_space(13)
AssertionError: 13 is not a string.
```

```
def before_space(s):
    """Returns the part of the string before the first space in s

    Precond: s a string with at least one space.
    Furthermore, s does not start with a space."""
```

5. [18 points] **Call Frames.**

Consider the following function definitions.

<pre>1 def muddle(seq): 2 """Returns seq with some stuff added 3 Pre: seq is a nonempty list""" 4 half = middle(seq,1) 5 copy = seq[:] 6 return half+copy 7</pre>	<pre>8 def middle(it,pos): 9 """Returns a slice of it at pos 10 Pre: it a nonempty list, pos > 0 an int""" 11 if pos > 1: 12 return it[pos:] 13 return it[:pos]</pre>
--	---

Assume `a = [1, 4, 5]` is a global variable referencing a list in the heap, as shown on the next page. Use the next two pages to diagram the evolution of the function call

```
a = muddle(a)
```

Diagram the state of the *entire call stack* for the function `muddle` 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 diagram on the next page

You should draw also the state of global space and the heap at each step. You can ignore the folders for the function definitions. Only draw folders for lists or objects. To help conserve time, you are allowed (and **encouraged**) to write “unchanged” if no changes were made to either a call frame, the global space, or the heap.

Last Name: _____

First: _____

Netid: _____

Call Stack

Global Space

The Heap

a

id1

id1

list

0

1

1

4

2

5

①

②

③

④

Last Name: _____ First: _____ Netid: _____

Call Stack

Global Space

The Heap

⑤

⑥

⑦

⑧

6. [22 points total] **Objects and Functions.**

As you are probably aware, angles can be measured in either degrees or radians. There are 180° in π , making conversion between the two easy. However, there is more than one way to specify degrees. We can specify degrees as decimals, like 75.3025° . Or we can break up that same value into degrees, *minutes* and *seconds* as follows: $75^\circ 18' 9''$ (the ' is for minutes and the " is for seconds).

There are 60 minutes to a degree (just like minutes and hours) and 60 seconds to a minute. If we need further accuracy, we add decimals to the seconds. For example, 75.302575° is the same as $75^\circ 18' 9.27''$. To implement this, we create an **Angle** class with the following attributes:

Attribute	Meaning	Invariant
degrees	the angle degrees	int value between 0 and 359 (inclusive)
minutes	1/60 of a degree	int value between 0 and 59 (inclusive)
seconds	1/60 of a minute	float value between 0.0 and 60.0 (excluding 60.0)

To make a new angle, we call the constructor function **Angle(d,m,s)**.

(a) [6 points] Implement the function below according to the specification.

```
def angle2decimal(angle):
    """Returns the decimal equivalent of the given angle

    Example if angle is the object Angle(75, 18, 9.27) then this
    function returns 75.302575 (the value returned is a float)

    Preconditions: angle is an Angle object"""
```

Last Name: _____ First: _____ Netid: _____

(b) [16 points] Implement the function below according to the specification.

```
def sub_angle(angle1,angle2):  
    """MODIFIES angle1 to be the result of subtracting angle2  
  
    This function is a procedure and does not return a value. If the angle  
    becomes negative, it wraps back around to 360.  
  
    Example: If angle1 is Angle(64,34,21) and angle2 is Angle(175,54,50),  
    then sub_angle(angle1,angle2) changes angle1 to Angle(248,39,31).  
  
    Preconditions: angle1 and angle2 are Angle objects"""
```