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.

Last Name: _____ First: _____ Netid: _____

# 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 |
|---|---|
| `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.count(s1)` | **Returns**: the number of times `s1` occurs in `s` |
| `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.rfind(s1)` | **Returns**: index of the first character of the LAST occurrence of `s1` in `s` (-1 if `s1` does not occur 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.isalnum()` | **Returns**: True if `s` is *not empty* and its elements are all letters or numbers; it returns False otherwise. |
| `s.islower()` | **Returns**: True if `s` is *has at least one letter* and all letters are lower case; it returns False otherwise (e.g. `'a123'` is True but `'123'` is False). |
| `s.isupper()` | **Returns**: True if `s` is *has at least one letter* and all letters are upper case; it returns False otherwise (e.g. `'A123'` is True but `'123'` is False). |
| `s.lower()` | **Returns**: A copy of `s` with all letters lower case. |
| `s.upper()` | **Returns**: A copy of `s` with all letters upper case. |

## List Functions and Methods

| Expression 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.count(y)` | **Returns**: the number of times `y` occurs in `x` |
| `x.index(y)` | **Returns**: index of the FIRST occurrence of `y` in `x` (an error occurs if `y` does not occur in `x`). |
| `x.index(y,n)` | **Returns**: index of the first occurrence of `y` in `x` STARTING at position n (an error occurs if `y` does not occur in `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`.

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.

- Global variables – variables assigned outside of function definition
- Local variables – variables assigned in a function definition
- Parameters – variables in a function header
- Attributes – variables in an object folder

(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.

A type cast is the conversion of a value from one type to another. A widening cast converts the type from one with less information to more information (e.g. int to float). An example is `float(2)` or `3/2.0`. A narrowing cast is the reverse – from more information to less. An example is `int(2.3)`.

(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)`

```
'Start first'
'Start second'
'In second try'
'In first except'
'Done first'
```

ii. `first(1)`

```
'Start first'
'Start second'
'In second except'
'Done second'
'In first try'
'Done first'
```

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"""

    # Verify the correct number of dashes
    c = s.count('-')
    if c == 0 or c > 2:
        return False

    # Find the first two strings
    pos1 = s.find('-')
    first = s[:pos1]
    second = s[pos1+1:]
    if c == 1:
        # Compare them
        return first == second

    # Find the third if as it must exist
    pos2 = s.find('-',pos1+1)
    second = s[pos+1:pos2]
    third = s[pos2+1:]
    # Compare them
    return first == second and second == third
```

Last Name: _____ First: _____ Netid: _____

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.*

There are many different possible answers to this question. Below are the different solutions we were thinking of. If you had (at least) six test cases that were close to the ones below, you got full credit. Otherwise, we checked if your test cases were *different enough*, and awarded you 2 points for each test.

| Input | Output | Reason |
|---|---|---|
| `seq=[]` | `[]` | Empty list |
| `seq=[1,'Hello',True]` | `[1]` | One int |
| `seq=[1,'Hello',2]` | `[1,2]` | Multiple ints, not adjacent |
| `seq=[1,2, 'Hello']` | `[1,2]` | Multiple ints, adjacent |
| `seq=[1,2,3]` | `[1,2,3]` | All ints |
| `seq=[True,False]` | `[]` | Non-empty, no ints |

(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'` |

Last Name: _____ First: _____ Netid: _____

**Note**: Write your answers on the next page

```
def romanize(n):
    """Returns the Roman numeral for n

    Example: romanize(74) returns 'LXXIV'

    Precond: 0 < n < 100 is an int"""
    tens = ''
    ones = ''
    if n >= 50:
        print('More than 50')       # TRACE
        tens = numeralL(n//10)
    elif n >= 10:
        print('More than 10')       # TRACE
        tems = numeralX(n//10)
    print('tens = '+repr(tens))     # WATCH

    ones = romanize1to9(n % 10)
    print('ones = '+repr(ones))     # WATCH
    return tens+ones


def romanize1to9(n):
    """Returns the Roman numeral for n

    Example: romanize1to9(5) returns 'V'

    Precond: 0 < n < 10 is an int"""
    # Combined TRACE and WATCH
    print('romanize1to9: n = '+repr(n))
    if n < 5:
        print('Less than 5')        # TRACE
        return romanize1to4(n)
    elif n < 9:
        print('Between 5 and 8')    # TRACE
        return 'V'+romanize1to4(n-5)
    else:
        print('Equal to 9')         # TRACE
        return 'IX'


def romanize1to4(n):
    """Returns the Roman numeral for n

    Example: romanize1to4(3) returns 'III'

    Precond: 0 < n < 5 is an int"""
    # Combined TRACE and WATCH
    print('romanize1to4: n = '+repr(n))
    values = ['I','II','III','IV']
    choiceI = values[n-1]

    # WATCH
    print('choiceI = '+repr(choiceI))
    return choiceI


def numeralL(n):
    """Returns Roman numeral for tens value

    The value n is the tens DIGIT of n
    Example: numeralL(6) returns 'LX'

    Precond: 5 <= n < 10 is an int"""
    # Combined TRACE and WATCH
    print('numeralL: n = '+repr(n))

    if n < 9:
        print('Less than 90')       # TRACE
        return 'L'+numeralX(n-5)
    else:
        print('Equals to 90')       # TRACE
        return 'XC'


def numeralX(n):
    """Returns Roman numeral for tens value

    The value n is the tens DIGIT of n
    Example: numeralX(3) returns 'XXX'

    When n is 0, it returns the empty
    string (to be compatible w/ numeralL)

    Precond: 0 <= n < 5 is an int"""
    # Combined TRACE and WATCH
    print('numeralX: n = '+repr(n))

    values = ['','X','XX','XXX','XL']

    choiceX = values[n]
    # WATCH
    print('choiceX = '+repr(choiceX))
    return choiceX
```

**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:**

This is a straight-forward misspelling error. The variable `tens` is mispelled as `tems` on **Line 14**, causing `romanize` to fall back to the original assignment on Line 5. To fix it, we rewrite Line 14 as the assignment

```
tens = numeralX(n//10)
```

```
>>> 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:**

The function `romanize1to4` is being called in such a way that the precondition is violated. That is because `romanize1to9` does not handle the case `n == 5` properly. We need to add the following code to **Line 33** of `romanize1to9` (just before the **elif**)

```
elif n == 5:
    return 'V'
```

```
>>> 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:**

This is *another* precondition violation of `romanize1to4`. But this time we see that the precondition of `romanize1to9` is also violated. So the problem is in the top level `romanize`. The problem is that we do not need to call `romanize1to9` at all if there is nothing in the ones position. So we should change **Line 17** to

```
if n % 10 > 0:
    ones = romanize1to9(n % 10)
```

(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."""

    assert type(s) == str, repr(s)+' is not a string.'
    assert ' ' in s, repr(s)+' has no spaces.'
    assert s[0] != ' ', repr(s)+' has an illegal space.'
```

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

Consider the following function definitions.

```
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
```

```
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]
```

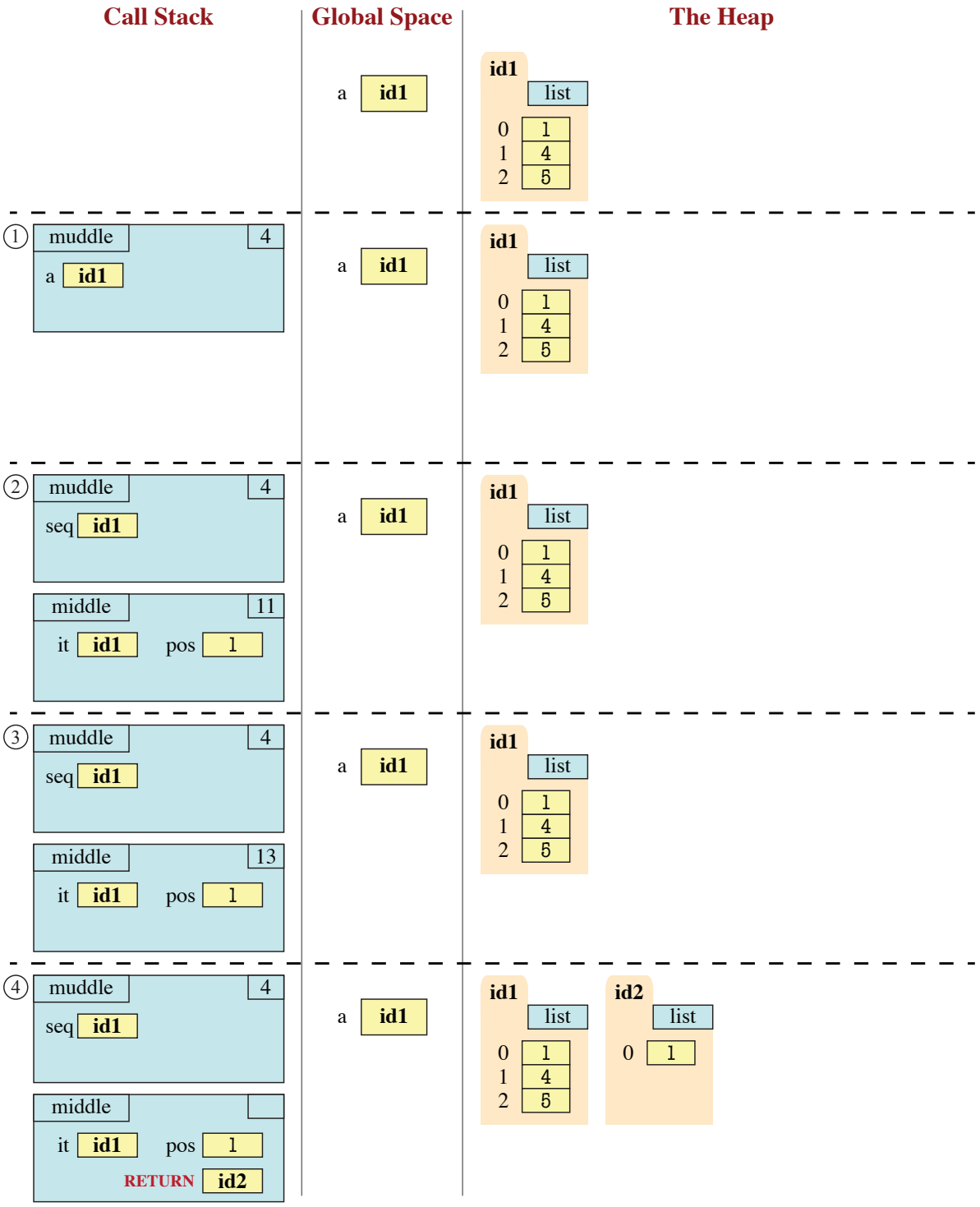
Assume `a = [1, 4, 5]` is a global variable referencing a list in the heap, as shown on the next page. Use the next two pages to diagram the evolution of the function call

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

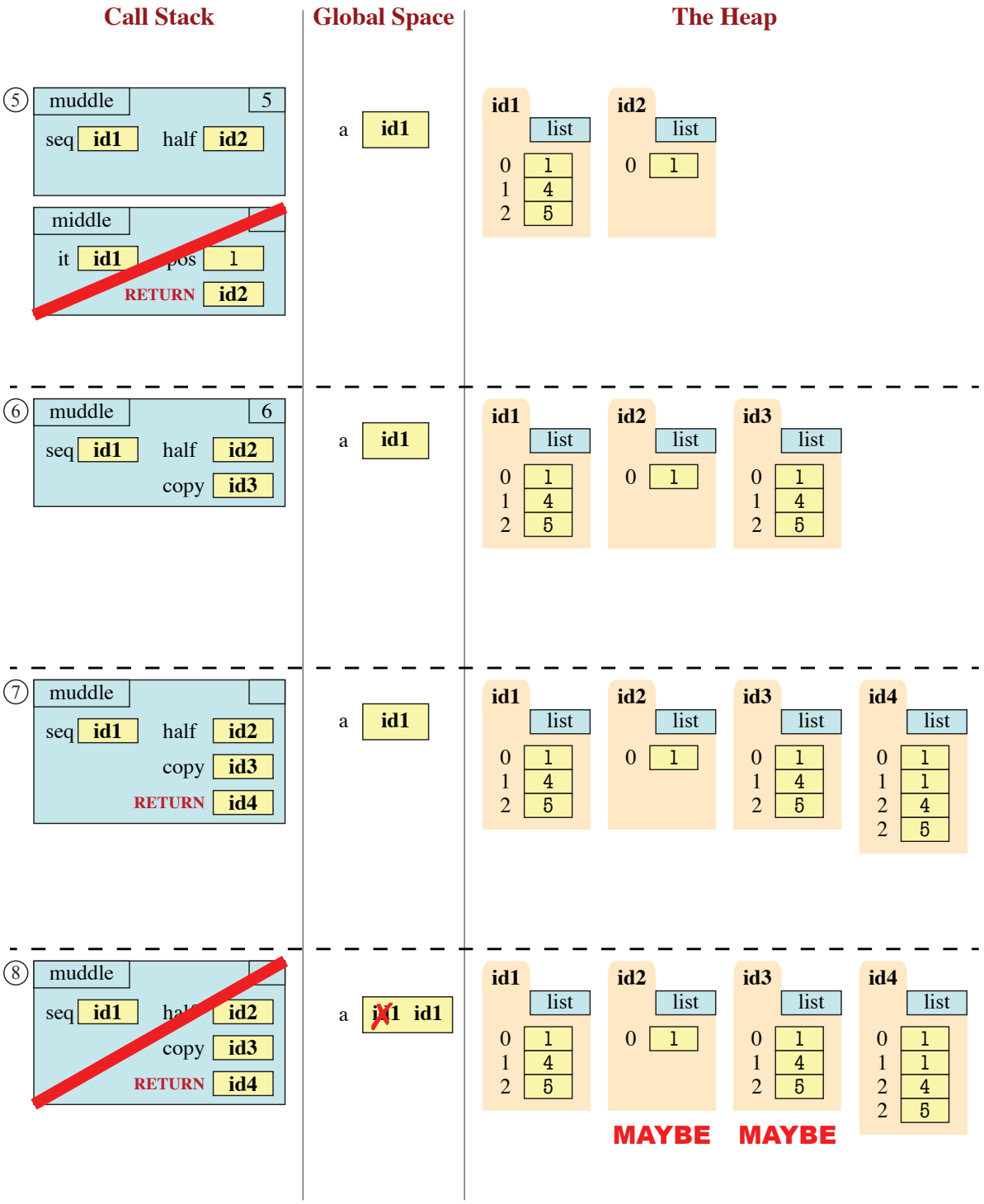
Diagram the state of the *entire call stack* for the function `muddle` when it starts, for each line executed, and when the frame is erased. If any other functions are called, you should do this for them as well (at the appropriate time). This will require a total of **eight** diagrams, not including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



Last Name: _____ First: _____ Netid: _____



Last Name: _____ First: _____ Netid: _____

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 $\pi$, 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 values in to degrees, *minutes* and *seconds* as follows: 75° 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° 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"""
    # Convert each attribute to a float in degrees
    d = float(angle.degrees)
    m = angle.minutes/60.0
    s = angle.seconds/(60.0*60.0)

    # Add them for the result
    result = d+m+s
    return result
```

(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"""

    # Subtract degrees, minutes, and seconds separately
    degrees = angle1.degrees-angle2.degrees
    minutes = angle1.minutes-angle2.minutes
    seconds = angle1.minutes-angle2.minutes

    # Adjust seconds if out of bounds
    if seconds < 0:
        seconds = seconds+60
        minutes = minutes-1

    # Adjust minutes if out of bounds
    if minutes < 0:
        minutes = minutes+60
        degrees = degrees-1

    # Adjust degrees if out of bounds
    if degrees < 0:
        degrees = degrees+360

    # Modify the attributes of angle1
    angle1.seconds = seconds
    angle1.degrees = degrees
    angle1.minutes = minutes
```