fullName: _____ andrewID: _____ section: ____

15-112 S26 Quiz10 Version A (Writeups Only)
Time: 15 Minutes

You must write your name on this paper and hand this back in immediately after the assessment. If we do not receive it immediately, you will receive a zero on the assessment. Do not unstaple any pages. All pages must be handed in intact.

Do not use your own scrap paper. You should not need it, but if you must absolutely have scrap paper, raise your hand and we will provide some. Write your andrewID clearly on it and hand it in with your quiz. We will not grade anything on scrap paper.

You may not ask questions during the quiz, except for English-language clarification questions. If you are unsure about a problem, take your best guess.

Before and during the quiz, you may not view any other notes, prior work, websites or resources, including any form of AI. You may not use calculators, phones, laptops, or any other devices. You may not communicate with anyone else except for current 112 TAs or faculty during the assessment. All syllabus policies apply.

You may not discuss this test with anyone else, even briefly, in any form, until we have released grades. Failure to abide by these rules may result in an academic integrity violation.

Do not use material we have not yet covered. Thus, do not use backtracking.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than the specified time noted above on this assessment.

Do not write on this handout (except your name, andrewId, and section) and be sure to hand this in in addition to your response sheet.

FR: ILS (IntegerLineSegment) class [100 pts]

Write the class ILS so the following test code passes. Do not hardcode any test cases. Your solution must work in general. However, you only have to implement the methods required to make the test code pass.

```
# When we create an IntegerLineSegment (abbreviated as ILS),
# we provide two possibly-float values, x0 and x1, where
# x0 <= x1.

# If x0 and x1 are integers, this results in an ILS from x0 to x1:
s1 = ILS(1, 3)
assert(str([s1]) == '[ILS(from 1 to 3)]')
assert(s1.x0 == 1)
assert(s1.x1 == 3)

# If x0 is not an int, use the largest int that is no larger
# than x0:
s2 = ILS(4.9, 6)
assert(str(s2) == 'ILS(from 4 to 6)')

# If x1 is not an int, use the smallest int that is no smaller
# than x1:
s3 = ILS(7, 8.1)
assert(str(s3) == 'ILS(from 7 to 9)')

# Two ILS instances are equal if they have the same x0 and x1:
s4 = ILS(7, 9)
assert(str(s4) == 'ILS(from 7 to 9)')
assert(s3 == s4)
assert(s3 != s2)
assert(s3 != 'do not crash here')

# Use s.contains(x) to test if the integer line segment s contains
# the possibly-float value x:
assert(s4.contains(8.5) == True) # 8.5 is between 7 and 9 (inclusive)
assert(s4.contains(9) == True)   # 9 is between 7 and 9 (inclusive)
assert(s4.contains(10) == False) # 10 is not between 7 and 9
```

```
# Use s.merge(t) to create a new Integer Line Segment r.
# The result r must contain every value in s or t,
# using the tightest possible integer bounds.
# This method should not modify s or t.
s = ILS(5, 6)
t = ILS(7, 8)

# When we merge these, we will get an ILS r where any value
# in s or t is also in r.  The smallest value in r must be 5 and
# the largest value in r must be 8, so:
r = s.merge(t)
assert(isinstance(r, ILS))
assert(str(r) == 'ILS(from 5 to 8)')

# Note that this contains all the values in s or t, but it can also
# contain some values that were NOT in s or t:
assert(r.contains(6.5) == True)

# verify that this was non-mutating
assert(s == ILS(5, 6))
assert(t == ILS(7, 8))

# Finally, ILS instances can be used in sets, so:
mySet = set()
assert(ILS(1, 2) not in mySet)
mySet.add(ILS(1, 2))
assert(ILS(1, 2) in mySet)
```

Do not write your response to the FR here.

Write your response on the separate handout.

BonusCT: These CTs are optional, and intended to be very challenging. They are worth very few points. Indicate what the following code prints. Place your answers (and nothing else) in the appropriate boxes **on the separate handout**. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

bonusCT1: [1pt]

```
def bonusCt1(s):
    def f(s, x):
        if s:
            c, s = s[-1], s[:-1]
            if c.isdigit():
                return 10*f(s, int(c)) + int(c) + x
            else:
                return f(s, x)
        else:
            return x
    return f(s, 0)

print(bonusCt1('What is 2+3-45?'))
```

bonusCT2: [1pt]

```
def bonusCt2(d, e):
    def f(n):
        r = 0
        while n-1:
            r += 1
            n = int(n**0.5)
        return r
    def g(n):
        if n < 5:
            return [n]
        else:
            n = f(n)
            return [n+e] + g(2**n)
    return g(2**d)

print(bonusCt2(40, 3))
```

Do not write your response to the BonusCTs here.
Write your response on the separate handout.