

fullName:_____ andrewID:_____ section:_____

15-112 F25

Practice Quiz1

This is the Practice Quiz1, part of Prep 2. In order to receive credit, you must simulate taking this as if it was an actual quiz, and you **must write your name on this paper and hand this back** in during lecture on 9/2. This Practice Quiz is graded based on completion rather than correctness, but if we do not receive it immediately, or if you have not demonstrated apparent effort, you will receive a zero on the assessment.

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

Do not use strings, loops, sets, dictionaries, recursion, or anything else disallowed in the original problem.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than 30 minutes on this assessment.

Code Tracing

CT1[10 pts]:

Indicate what the following code prints. Place your answer (and nothing else) in the box below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

```
import math

def ct1(x, y):
    print(math.ceil(x))
    print(math.floor(-x))
    print(type(True) == isinstance(int(y), int))

print(ct1(3.14, '5'))
```

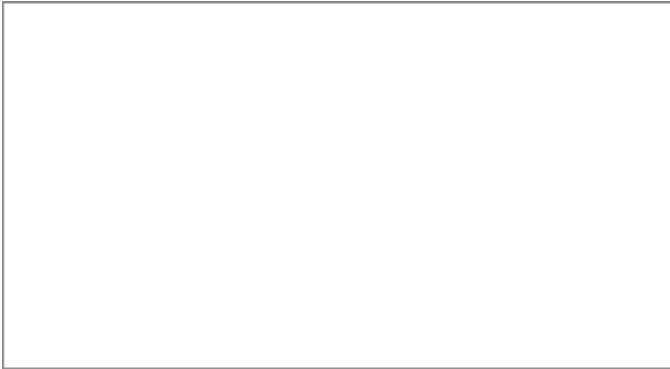


CT2[10 pts]:

Indicate what the following code prints. Place your answer (and nothing else) in the box below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

```
def ct2(x):  
    print((x > 0) or (x/0 > 0))  
    print((x > 0) and (x/0 > 0))  
    return 42
```

```
print(ct2(1))
```



CT3[10 pts]:

Indicate what the following code prints. Place your answer (and nothing else) in the box below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

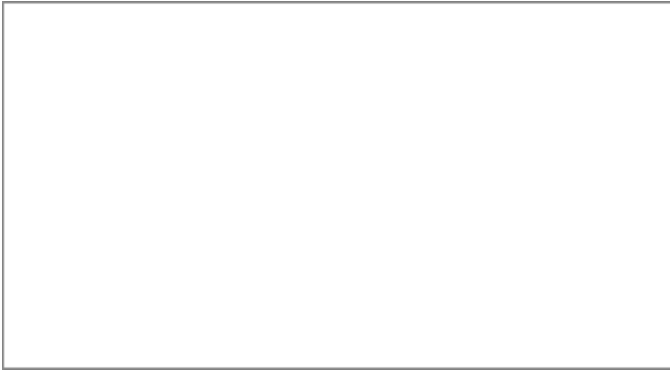
```
def ct3(x, y):  
    if x%10 == x//10:  
        print('A', 10*y-x/10)  
    x, y = y, x  
    x **= 2  
    if x > y:  
        print('B', x, bool(y))  
    else:  
        print('C', -x, not bool(y))  
    return isinstance(x, float)  
  
print(ct3(11,3))
```



CT4[10 pts]:

Indicate what the following code prints. Place your answer (and nothing else) in the box below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

```
def f(x):  
    return x+x%10  
  
def g(x):  
    return f(x) + f(x+1)  
  
def ct4(x):  
    x = f(x)  
    return f(g(x))  
  
print(ct4(5))
```



Free Response

Your functions should work generally for the kinds of inputs specified in the problem statement, and we may test your code using additional test cases. In graded quizzes, we will manually grade both of these problems for partial credit if you do not pass all the test cases.

FR1[30pts]: `distanceToMultipleOf10(n)`

Write the function `distanceToMultipleOf10(n)` that takes a possibly-negative int `n` and returns the distance (as an int) to the multiple of 10 nearest to `n`.

Here are some test cases for you:

```
assert(distanceToMultipleOf10(60) == 0)
assert(distanceToMultipleOf10(61) == 1)
assert(distanceToMultipleOf10(65) == 5)
assert(distanceToMultipleOf10(66) == 4)
assert(distanceToMultipleOf10(1234567) == 3)
assert(distanceToMultipleOf10(-1234567) == 3)
```

Write your answer on the following page



FR2[30pts]: isNearlyXor(f)

Background: the function `xor(x, y)` takes two booleans, `x` and `y`, and returns `True` if `x` and `y` are not the same, and `False` otherwise. Here is a truth table for `xor`:

x	y	xor(x,y)
True	True	False
True	False	True
False	True	True
False	False	False

We will say that a function is "nearly xor" (a coined term) if its truth table differs from `xor` in exactly one row.

With that in mind, write the function `isNearlyXor(f)` that takes a function `f`, where `f` takes two boolean values and returns a boolean value, and returns `True` if `f` is nearly xor, and `False` otherwise.

Here are some test cases for you:

```
def f1(x, y): return ((x and not y) or (not x and y)) # this is xor
def f2(x, y): return (x or y)
def f3(x, y): return (x and y)
assert(isNearlyXor(f1) == False) # "xor" differs from "xor" in 0 rows
assert(isNearlyXor(f2) == True)  # "or" differs from "xor" only in the first row
assert(isNearlyXor(f3) == False) # "and" differs from "xor" in 3 rows
```

Write your answer on the following page

