

**15-112 Spring 2020 Quiz 2**

**Up to 20 minutes. No calculators, no notes, no books, no other paper, no computers.**

**No strings, lists, string or list indexing, or recursion**

**You may call `almostEqual(x, y)` and `roundHalfUp(d)` without writing them. Write everything else!**

**1. Code Tracing [20pts]**

Indicate what the following code prints. Place your answer (and nothing else) in the box to the right.

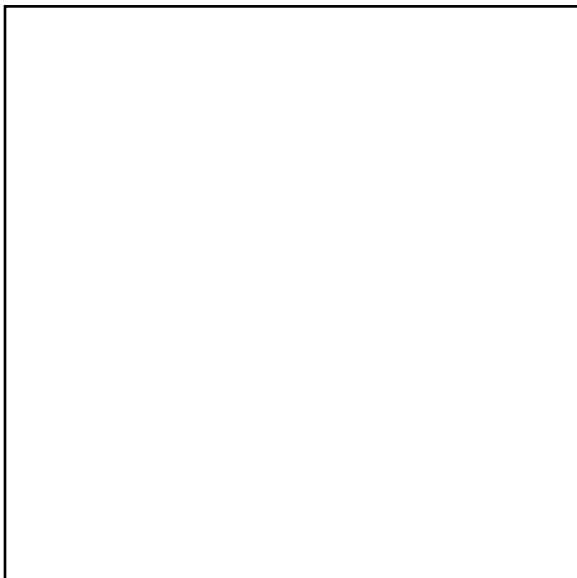
```
def ct1(x, y):  
    for i in range(x):  
        for j in range(i, y):  
            if (i + j) % 3 == 0:  
                print(i, j)  
        if i < 2:  
            print("whee!")  
    return i + j  
  
print(ct1(3, 5))
```



**2. Graphics Code Tracing [25pts]**

Sketch what the following code draws. Your sketch should be close, but not necessarily pixel-perfect.

```
import basic_graphics  
  
def drawCT(canvas, w, h):  
    a = 300  
    canvas.create_polygon(0, h/2, w, h/2, w/4, h/4,  
        fill = "", outline = "black")  
    canvas.create_oval(w/2, h/2, a, a)  
    canvas.create_text(a, a, text="Yay", anchor="w")  
  
basic_graphics.run(width=400, height=400, drawFn=drawCT)
```



**3. Free Response: countPalNumbers(n) [40pts]**

Write the function `countPalNumbers(n)`, which takes a positive integer `n` and returns the number of palindrome numbers (pal for short) that exist between 1 and `n` (inclusive). A palindrome number is an int that is the same forwards as backwards; for example, 121 is a palindrome number, as is 7. 1231 is not a palindrome number, as it is not equal to 1321.

**Note: you may not use strings in this problem!!** A solution that uses strings will receive 0 points.

# Here are some test cases:

```
assert(countPalNumbers(1)==1)
```

```
assert(countPalNumbers(5)==5)
```

```
assert(countPalNumbers(10)==9)
```

```
assert(countPalNumbers(100)==18)
```

**4. Reasoning over Code [15pts]**

Find an argument for the function `rc1(n)` that makes it return `True`. Place your answer (and nothing else) in the box to the right.

**Hint: Start by trying a 4 digit number like 1234, and see what values `a` and `b` hold!**

```
def rc1(n):
    assert((n >= 1000) and isinstance(n, type(42)))
    a = b = count = 0
    while (n > 0):
        a = 10*a + n%10
        b = 10*b + (n//10)%10
        n //= 10
    for i in range(a, b):
        if (i % 10 == 0):
            count += 1
    return ((a > 75) and
            ((a % 10) * (b % 10) == 8) and (count == 2))
```

n =

**5. Bonus CT (This problem is optional! Answer in the box to the right.) [3pts]**

```
def bonusCt(n):
    (a,b,c) = (0,1000, 100)
    while (c < 1000):
        for x in range(a, b, c):
            (a,b,c) = (a+1, b-1, c+50)
    return a-n
print(bonusCt(2))
```