Name: _____ Section: _____ Andrew Id : _____

15-112 Summer-1 2017 Quiz 1

***Up to 45 minutes. No calculator, no notes, no books, no computers. Show your work!**
No strings, list, or recursion!

1.**Code Tracing** [20 pts]: Indicate what these print. Place your answers (and nothing else) in the boxes below the code.

```
y = 100
def ct1(x, y):
    for i in range(1, x, 3):
        if (i % 2 == 0):
            print("A:", i, end = " ")
        elif (i%10 == y%10): print("C:", i, y, end = "")
        if ((i**(0.5)) % 1 == 0):
            print("B:", i, end = " ")
        print()
        y += 1

ct1(10, 5)
print(y)
```

```
def ct2(x):
    y = 5
    for z in range(x+y, 10, -2):
        if (z > 20): print ('z', z, end = ' ')
        elif (z//10 == z%10): print ('z', z, end = ' ')
        for w in range(z, 20, y):
            y += 1
            if (w % 10 == 8): print('w', w, end = ' ')

print(ct2(8))
```

2. **Reasoning Over Code** [20 pts]: Find an argument for the following function that makes it return True. Place your answers (and nothing else) in the boxes below the code.

```
def f(x, y):
    assert((type(x) == int) and (type(y) == int) and (100 > x > y > 0))
    if (x + y == 50):
        z = 0
    else:
        z = 123
    while (x//10 != y//10):
        x -= 10
        z += 1
    return (x == z == 4)
```

x =	y =

```
def g(x, y):
    assert(type(x) == type(y) == int)
    assert(100 > x > y > 0)
    s = 0
    while (x > y):
        y += 3
        x -= 2
        s += 1
    return (s == 10)
```

x =	y =

3. **Free Response 1: formNumberFromOddDigits(n)** [30 pts]: Write the function formNumberFromOddDigits(n) which you may abbreviate as f(n) which takes a possibly-negative int value n and returns a new integer of the same sign that only contains the odd digits of the original input. So for example f(9421) returns 91, since the only odd digits in 9421 are 9 and 1. If the number contains no odd digits, then return 0.

4. **Free Response 2: nthWithTwoOnes(n)** [30 pts]: Write the function nthWithTwoOnes(n) that takes a non-negative int n and returns the nth positive integer that contains exactly two ones. Thus, 11, 211, and 121 are such numbers, but 111 is not. Remember we always start counting from 0, so nthWithTwoOnes(0) = 11.