Question 3

Similar to the combining part of a merge sort except we only consider the larger half of the array. Each iteration, we will compare random pairs of apples and discard the lighter one. Total number of apples is 1024 which is conveniently a power of two so we will always have a pair until there is only one apple left. If the number wasn't a perfect power, we would just leave the leftovers for the next iteration. After we do 512 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 1023 comparisons, we will have the largest apple.

However, to get the second heaviest apple, we cannot just take our 2nd last one. This is because we only take the heavier apple on each iteration and discard the lighter one. There is a possibility that in the middle, we compared the heaviest with the second heaviest and ended up discarding it. The workaround to this issue is to keep track of which apples each one weighed against. This means when we are down to our heaviest apple, we will have all the other 9 apples it previously weighed against. Going through these 9 apples to find the heaviest will result in the second heaviest apple overall.

SAMPLE CODE

```
#!/usr/bin/python3
import random

def main():
    # TESTING

    print(q3().weight) #  should be 1023


def q3():
    # apples with their own weights which we do not know
    class Apple:
        def __init__(self, weight):
            self.weight = weight
            self.previousWeighs = []


    # our 'scale' which returns heavier/ligher apple
    class Scale:
        def __init__(self):
            self.nUsed = 0

        # returns the (heavier, lighter) apple
        def weigh(self, apple1, apple2):
            self.nUsed += 1
            return (apple1, apple2) if apple1.weight > apple2.weight else (apple2, apple1)

    scale = Scale()
    apples = [Apple(i + 1) for i in range(1024)]
    random.shuffle(apples)

    # merge sort and only consider the larger half each time until we have 2 apples left
    while len(apples) > 2:
        remaining = []

        for i in range(1, len(apples), 2):
            # compare each pair
            apple1 = apples[i]
            apple2 = apples[i - 1]
            heavier, lighter = scale.weigh(apple1, apple2)

            # keep track of all comparisons in case we weigh with second heaviest
            # we will check this at the end
            remaining.append(heavier)
            heavier.previousWeighs.append(lighter)

        apples = remaining

    # most of the time this will be heaviest and second heaviest
    heaviest, second = scale.weigh(apples[0], apples[1])

    # however, sometimes we get unlucky and compare heaviest and second heaviest
    # go through all the comparisons again to get second heaviest
    for apple in heaviest.previousWeighs:
        second, _ = scale.weigh(second, apple)

    # make sure we use scale 1032 times
    assert(scale.nUsed <= 1032)

    # and we found the correct one
    assert(second.weight == 1023)

    return second

if __name__ == '__main__':
    main()
```