Computer Science 134C – Fall 2018 **Anonymous ID:**

Duane A. Bailey

Homework 4 – *Due: Wednesday, October 10*

Please turn in answers to the following questions this Friday, in class.
For the next few questions, we will think about the implications of variables (like `alice` and `bob`) being references to immutable and mutable objects. Beside each `print:`, indicate what is printed.

1a.
```
alice = [ 3, 5, 11 ]  # some odd numbers
bob =   [ 3, 5, 11 ]  # some primes
print(alice is bob)   # prints:
alice.append(7)       ###
print(alice)          # prints:
print(bob)            # prints:
```

Explain what is happening to `alice` and `bob` (if anything) on the statement marked ###.

1b.
```
alice = bob = [ 5, 3, 11, ] # some odd and prime numbers
print(alice is bob)   # prints:
sorted(alice)         ###
print(alice)          # prints:
print(bob)            # prints:
bob.sort()            ###
print(alice)          # prints:
print(bob)            # prints:
```

Explain what is happening on the statements marked ###.

1c.
```
alice = bob = 'odd numbers'
print(alice is bob)   # prints:
bob.replace('odd','prime') ###
print(bob)            # prints:
print(alice)          # prints:
alice = alice.replace('odd','uneven') ###
print(alice)          # prints:
print(bob)            # prints:
```

Explain what is happening on the statements marked ###.

Suppose we're interested in building a dominoes game. Two operations we would need to perform are (1) the construction of the set of tiles and (2) the shuffling of the tiles themselves. Let's assume that a single domino is represented as a pair of numbers between 0 and 6 where the first element of the pair is never greater than the second. For example, `(0,6)` is a valid domino. A set of dominoes is simply all unique combinations of pairs of values.

2a. Here is some code that is supposed to generate a set of dominoes. Unfortunately, it has some logical errors. Edit the code to remove the errors while maintaining the general spirit of the logic. Please do not use list comprehensions.

```
def tiles(pips=6):
    "Returns a set of domino tiles.
     Each tile is a unique combination of
     pips between 0 and 'pips'."
    results = []
    for front in range(pips):
        for back in range(0, pips):
            tile = tuple(front,back)
            results += list(tile)
    return results
```

2b. The following code shuffles a list of n tiles by performing a series of $\frac{n}{2}$ exchanges. Each exchange randomly selects two items of the list and then swaps the items. Again, the code is not quite right. Please make changes that reflect the desired logic without using comprehensions.

```
def shuffle(pile):
    """Shuffles a list of items."""
    n = len(pile)
    for _ in range(0,n/2):
        left = randint(0,n-1)
        right = randint(left,n)
        temp = pile[left]
        pile[right] = pile[left]
        pile[left] = temp
    return pile
```

2c. Extra: Compare the two ways we've seen to shuffle items of a list this week: here, by item exchange, in lab, by random draw.

⋆