CS 111 first name: Xuan last name: Hu

Quiz 4 BU ID: U26611932

1. What is the output of each of the following code fragments? (4 points)

a)

```
for i in [2, 3]:
    for j in range(1, i):
        print(i, j)
    print(i, j)
```

# put the output below:

```
2 1
2 1
3 1
3 2
3 2
```

c)

```
def foo(a, b):

    for i in range(len(a)):
        a[i] *= 2
        b += 1
    print(a[3], b)

a = 42
b = [1, 2, 3, 4]
c = b

foo(c, a)
print(b[3], a)
```

# put the output below:

```
8 46
8 42
```

2. (5 points) Write a function `scale(grid, scalar)` that takes a 2-D list `grid` and a `scalar` (a numeric value), and multiplies each element of `grid` by the value `scalar`. The result of the function will be to modify each element of the `grid` to hold the corresponding scaled value. The function should change `grid`, but it should *not* return anything. For example:

```
>>> m = [[-2, 3, 4, -1], [-5, 0, -7, 6], [0, -3, 7, -9]]
>>> scale(m, 3)
>>> print(m)
[[-6, 9, 12, -3], [-15, 0, -21, 18], [0, -9, 21, -27]]
```

n

2.
```
def scale(grid, scalar):
    """
        takes a 2-D list grid and a scalar, and multiplies
        each element of grid by the value scalar.
    """
    for r in range(len(grid)):
        for c in range(len(grid[0])):
            grid[r][c] *= scalar
```

3.
```
def sum(a,b):
    """takes 2-D lists a and b, and return a new 2-D list
       in which each element is the sum of the element
       in the corresponding position.
    """
    ans = [[0 for col in range(len(a[0]))] for row in
                                          range(len(a))]
    for r in range(len(a)):
        for c in range(len(a[0])):
            ans[r][c] = a[r][c] + b[r][c]
    return ans
```

4.

a) `len(drink_orders)`

b) `drink_orders['Ellie']`

c) `drink_orders['Aaron'] = 'iced coffee'`

d)
```
for key in drink_orders:
    print(key, drink_orders[key])
```

3. (5 points) Write a function `sum(a, b)` that takes as parameters two 2-D lists a and b, and returns a new 2-D list in which each element is the sum of the elements in the corresponding positions in the parameter lists a and b. For example:

```
>>> a = [[1, 2, 3], [4, 5, 6]]
>>> a = [[2, 2, 2], [3, 3, 3]]
>>> c = sum(a, b)
>>> print(c)
[[3,4,5],[7,8,9]
```

For simplicity, you may assume that 2-D lists a and b have the same dimensions.

4. (6 points) Consider the following code fragment, which creates a dictionary of names and drink orders (represented as strings).

```
drink_orders = {'Caleb': 'Sprite', 'Ellie': 'Gatorade red', 'Jacob':
'Water'}
```

a) Write expression to obtain the number of elements in this dictionary:

b) Write expression to obtain Ellie's drink order:

c) Write expression to add an entry for Aaron's drink (iced coffee) into this dictionary:

d) Write a code fragment that will iterate over all entries in this dictionary and print out the person's name and drink order, one per line, separated by a space: