CSCA08H Worksheet: Dictionaries

1. Consider this code:

```
name_to_binomial = {'human': 'Homo sapiens',
                    'dog': 'Canis familiaris',
                    'narwhal': 'Monodon monoceros'}
```

Circle the expressions that evaluate to `True`.

(a) `'dog' in name_to_binomial` — True

(b) `'Canis familiaris' in name_to_binomial` — False: not a key

(c) `name_to_binomial[0] == 'human'` — Error! 0 not a key

(d) `len(name_to_binomial) == 3` — True: len is number of key/value pairs

(e)
```
name_to_binomial == {  'dog': 'Canis familiaris',
                       'narwhal': 'Monodon monoceros',
                       'human': 'Homo sapiens'}
```
True: no ordering, equal key/value pairs

2. Suppose we begin with the following assignment statement:

```
animal_to_locomotion = {'fish': ['swim'],
                        'kangaroo': ['hop'],
                        'frog': ['swim', 'hop']}
```

What is the result of each piece of code below? (Consider each piece of code independently.)

(a) `animal_to_locomotion['human'] = ['swim', 'run', 'walk', 'airplane']` — add new key/value pair

Error? (Yes / No)

`animal_to_locomotion`: {'fish': ['swim'], 'kangaroo': ['hop'], 'frog': ['swim', 'hop'], 'human': ['swim', 'run', 'walk', 'airplane']}

(b) `animal_to_locomotion['orangutan'].append('brachiate')`

Error? (Yes / No) — 'orangutan' not a key: KeyError

`animal_to_locomotion`: {'fish': ['swim'], 'kangaroo': ['hop'], 'frog': ['swim', 'hop']}

(c) `animal_to_locomotion['kangaroo'].append('airplane')`

Error? (Yes / No)

`animal_to_locomotion`: {'fish': ['swim'], 'kangaroo': ['hop', 'airplane'], 'frog': ['swim', 'hop']}

modify existing value

(d) `animal_to_locomotion['frog'] = ['tapdance']`

Error? (Yes / No)

`animal_to_locomotion`: {'fish': ['swim'], 'kangaroo': ['hop'], 'frog': ['tapdance']}

update value for key 'frog'

(e)
```
animal_to_locomotion['dolphin'] = animal_to_locomotion['fish']
animal_to_locomotion['fish'].append('surprise!')
```

Error? (Yes / No)

`animal_to_locomotion`: {'fish': ['swim', 'surprise!'], 'kangaroo': ['hop'], 'frog': ['swim', 'hop'], 'dolphin': ['swim', 'surprise!']}

the value for key 'fish' is the same list / same object in memory as value for key 'dolphin'

3. The express checkout is for grocery orders with 8 or fewer items. Complete the examples in the docstring and then complete the function body.

```
EXPRESS_MAX = 8

def express_checkout(product_to_quantity: Dict[str, int]) -> bool:
    """Return True if and only if the grocery order in product_to_quantity
    qualifies for the express checkout. product_to_quantity maps
    products to the numbers of those products in the grocery order.

    Precondition: all values in product_to_quantity are non-negative.

    >>> express_checkout({'banana': 3, 'soy milk': 1, 'peanut butter': 1})
    True
    >>> express_checkout({'banana': 3, 'soy milk': 1, 'twinkie': 7})
    [                                        ]

    >>> express_checkout( [                                        ] )
    [                                        ]

    >>> express_checkout( [                                        ] )
    [                                        ]

    >>> express_checkout( [                                        ] )
    [                                        ]
```

4. As part of a study funded by a major shoe company, we crouched at the finish line of the Boston marathon and kept an ordered list of the shoes that we saw as they passed the finish line. Write a function that, given such a list, will return a dictionary mapping shoe companies to a list of the placements achieved by runners wearing shoes made by those companies.

```
def build_placements(shoes_at_finish_line: List[str]) -> Dict[str, List[int]]:
    """Return a dictionary built from the information in
    shoes_at_finish_line, where each key is a brand and each value is
    a list of placements by people wearing shoes made by that
    company. shoes_at_finish_line contains shoe brands in order, in
    which they appeared at the finish line.

    >>> result = build_placements(['Saucony', 'Asics', 'Asics', 'NB',
    ...                            'Saucony', 'Nike', 'Asics', 'Adidas',
    ...                            'Saucony', 'Asics'])
    >>> result == {'Saucony': [1, 5, 9], 'Asics': [2, 3, 7, 10], 'NB': [4],
    ...                         'Nike': [6], 'Adidas': [8]}
    True
```

Add more examples:

"""

Add function body: