CSCI 145 Quiz 6 practice questions on recursion, tuples, and dictionaries

1. Consider this code:

```
def peel(word):
    if len(word) == 0:
        print("*")
    else:
        print(word[0])
        peel(word[1:])
        print(word)
peel("hat")
```

a) Write the exact output.

b) How many times is `peel` called, counting `peel("hat")`?

_____

c) Which call prints *?

_____

2. Each line is the recursive case of a function. Write Tail if it is tail recursion, or Pending if it has a pending operation.

```
return find(nums[1:], target)
```
_____

```
return len(words[0]) + letters(words[1:])
```
_____

```
return dashes(n - 1) + "-"
```
_____

```
return last(word[1:])
```
_____

3. Consider this code:

```
def stretch(word):
    if len(word) == 0:
        return ""
    return word[0] + word[0] + stretch(word[1:])
```

a) What does `stretch("abc")` return?

_____

b) Fill in the blanks to show how `stretch("abc")` is computed.

```
"aa" + (________ + (________ + ________))
```

4. Consider this code:

```
import turtle as t
def hook(length):
    if length > 15:
        t.forward(length)
        t.left(90)
        hook(length // 2)
        t.right(90)
        t.backward(length)
hook(80)
```

a) How many times is `hook` called, counting `hook(80)`?

_____

b) Which runs first: `t.backward(80)` or `t.backward(20)`?

_____

c) Where does the turtle end? Circle one.

A. At the end of the last line it drew
B. Where it started, facing left
C. Where it started, facing the same way
D. 20 steps from where it started

5. Write a recursive function `no_zeros(nums)`, without a loop. If `nums` is empty, it returns an empty list. Otherwise, it returns a new list with the numbers in `nums` that are not 0, in the same order. Do not include a docstring. For example:

```
>>> no_zeros([0, 4, 0, 7])
[4, 7]
>>> no_zeros([0, 0])
[]
>>> no_zeros([])
[]
```

6. Suppose `dims = (12, 5, 30)`. Write the value of each expression.

`dims[1]`	_____	`dims[-1]`	_____	`dims[:2]`	_____
`len(dims)`	_____	`5 in dims`	_____	`dims[1] * 2`	_____

7. Answer each part. For b) and c), circle one answer.

a) Write the exact output.

```
a, b, c = 1, 2, 3
a, b = b, a
c, a = a + c, c
print(a, b, c)
```

Output

b) What happens when this runs?

```
dims = (12, 5, 30)
dims[0] = 15
print(dims)
```

A. It prints `(15, 5, 30)`
B. A `TypeError`
C. It prints `(12, 5, 30)`
D. It prints `15`

c) What happens when this runs?

```
ages = {"Ana": 19}
print(ages["Cy"])
```

A. It prints `0`
B. It prints `None`
C. It prints nothing
D. A `KeyError`

8. Write the exact output of each program, one printed line per line.

a)

```
def split_at(word, i):
    return word[:i], word[i:]
parts = split_at("sunset", 3)
front, back = parts
print(parts)
print(back, front)
```

b)

```
prices = {"tea": 3, "pie": 5, "jam": 4}
for item, price in prices.items():
    if price > 3:
        print(item, price * 2)
print(sum(prices.values()))
```

9. Consider this code:

```
def restock(stock, item):
    stock[item] = stock.get(item, 0) + 5

inv = {"pen": 4, "cup": 2}
inv["cup"] += 1
result = restock(inv, "mug")
print(len(inv))
print(inv["mug"], inv.get("ink", 0))
print("pen" in inv, 4 in inv)
print(result)
```

a) Write the exact output.

b) What is `inv` at the end?

_____

10. Write a function `count_lengths(words)`. It returns a dictionary that maps each word length to how many words in `words` have that length. If `words` is empty, it returns an empty dictionary. Do not include a docstring. For example:

```
>>> count_lengths(["a", "ox", "hi"])
{1: 1, 2: 2}
>>> count_lengths([])
{}
```