Diagnostic Quiz Record

1. Problem1. Catch that Bug!

You may assume that digit_counter will only be passed a positive integer for its second parameter.

1.1. Problem1 debug1

```
def digit_counter(f, item):
    counter = 0
    while item >= 0:
        if f(item % 10):
            counter += 1
        item = item // 10
    return counter
```

1.2. Problem1 debug2

```
def digit_counter(f, item):
    if item < 10 and f(item):
        return item
    if f(item % 10):
        return 1 + digit_counter(f, item // 10)
    return digit_counter(f, item // 10)
```

1.3. Problem1 debug3

```
def digit_counter(f, item):
    def helper(x, sofar):
        if x > item:
            return sofar
        last = (item // x) % 10
        return helper(x * 10, sofar + f(last))
    return helper(0, 0)
```

> solution

debug1的`item>=0`去掉`=`

debug2的`return item`改为`return 1`

debug3的`helper(0,0)`改为`helper(1,0)`

2. Problem2. Applications are Closed

You may assume that x is always less than the second argument of the function.

3. Problem3. Camel Sequence

```
def is_camel_sequence(n):
    """
    >>> is_camel_sequence(15263) # 1 < 5, 5 > 2, 2 < 6, 6 > 3
    True
    >>> is_camel_sequence(98989)
    True
    >>> is_camel_sequence(123) # 1 < 2, but 2 is not greater than 3.
    False
    >>> is_camel_sequence(4114) # 1 is not strictly less than 1
    False
    >>> is_camel_sequence(1)
    True
    >>> is_camel_sequence(12)
    True
    >>> is_camel_sequence(11)
    False
    >>> is_camel_sequence(11910986)
    False
    """
    def helper(n, thank):
        if _____________:
        #      (a)
            return True
        elif thank:
            return _________________ and helper(_____________)
            #            (b)                        (c)
        else:
            return _________________ and helper(_____________)
            #            (d)                        (e)
    return _____________ or _____________
```

Problem2 soluton

```
def is_camel_sequence(n):

    def helper(n, thank):
        if n//10==0:
        #      (a)
            return True
        elif thank:
            return n % 10 < n // 10 % 10 and helper(n//10,False)
            #            (b)                        (c)
        else:
            return n % 10 > n // 10 % 10 and helper(n//10,True)
            #            (d)                        (e)
    return helper(n,True) or helper(n,False)
```