CSCA08H Worksheet: Nested Lists and Loops

1. Consider this code:

```
data = [['a', 'b'], [3, 4], ['cat', 'mouse', 'elephant']]
sublist = data[2]
```

(a) What is the value of each of the following expressions after the above code is executed?

Expression	Value
`data[2]`	['cat', 'mouse', 'elephant']
`data[2][1]`	'mouse'
`len(data[2])`	len(['cat', 'mouse', 'elephant']) ==> 3
`len(data[2][1])`	len('mouse') ==> 5
`data[1]`	[3, 4]
`data[1][0]`	3
`sublist[0]`	['cat', 'mouse', 'elephant'][0] ==> 'cat'

(b) If we add the following as a **third** instruction, what are the values of `data` and `sublist` after it is executed:

```
data[2][1] = 'dog'
```

`data`	[['a', 'b'], [3, 4], ['cat', 'dog', 'elephant']]
`sublist`	['cat', 'dog', 'elephant']

(c) If we add the following as a **fourth** instruction, what are the values of `data` and `sublist` after it is executed:

```
sublist.append('rat')
```

`data`	[['a', 'b'], [3, 4], ['cat', 'dog', 'elephant', 'rat']]
`sublist`	['cat', 'dog', 'elephant', 'rat']

(d) If we add the following as a **fifth** instruction, what are the values of `data` and `sublist` after it is executed:

```
sublist = ['new', 'list']
```

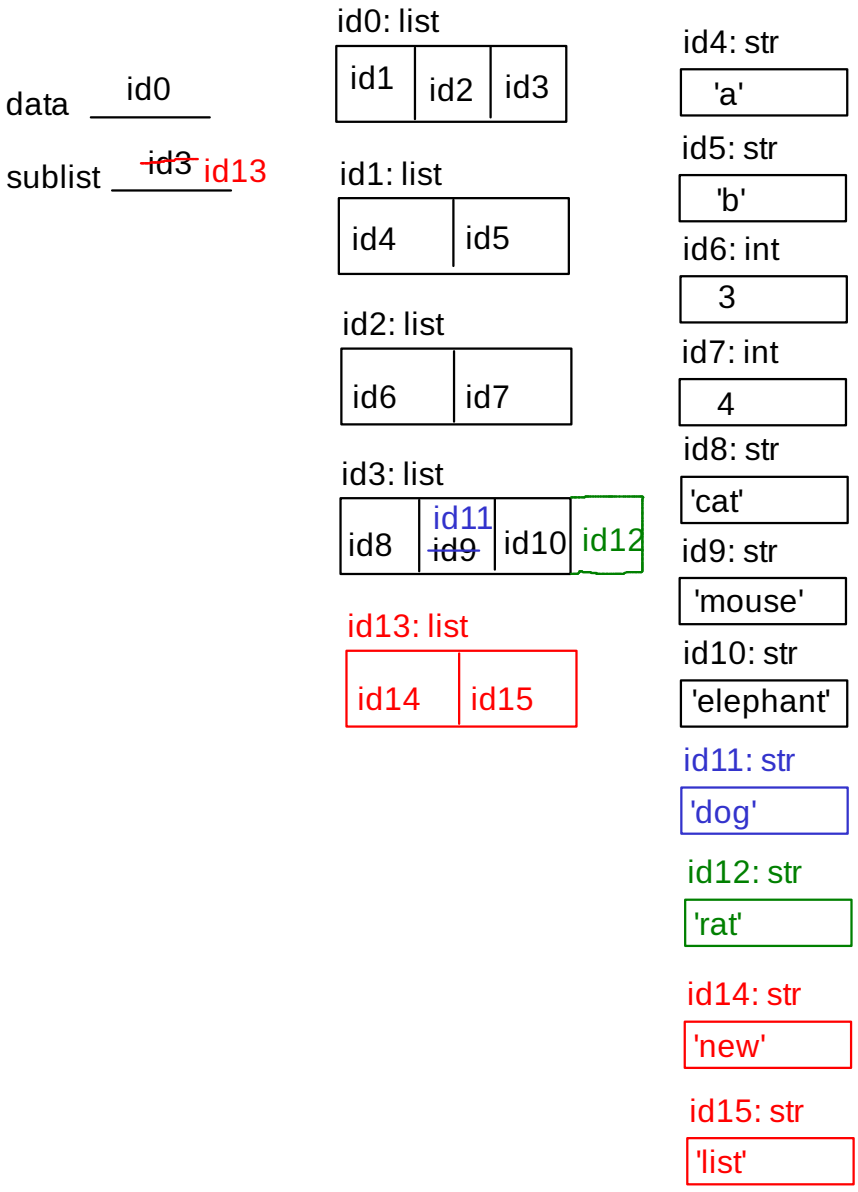
`data`	[['a', 'b'], [3, 4], ['cat', 'dog', 'elephant', 'rat']]
`sublist`	['new', 'list']

```
data = [['a', 'b'], [3, 4], ['cat', 'mouse', 'elephant']]
sublist = data[2]

data[2][1] = 'dog'

sublist.append('rat')

sublist = ['new', 'list']
```



2. What is the value of `lst` after each of the following programs is executed?

```
lst = []
for i in range(4):
  lst = lst + [i]
```

i: 0,1,2,3

lst	[0, 1, 2, 3]

```
lst = []
for i in range(4):
  lst.append([i])
```

lst	[[0], [1], [2], [3]]

```
lst = []
for i in range(4):
  lst = lst + [[i]]
```

lst	[[0], [1], [2], [3]]

```
lst = []
for i in range(4):
  lst.extend([i])
```

lst	[0, 1, 2, 3]

```
lst = [0, 1, 2, 3]
lst[-1] = [3, 4, 5]
```

lst	[0, 1, 2, [3, 4, 5]]

```
nums = [1, 2, 3]
lst = ['a', 'b', nums, 'c']
```

lst	['a', 'b', [1, 2, 3], 'c']

3. Complete the examples in the docstring and then the function body.

```
def digital_sum(nums_list: List[str]) -> int:
    """Return the sum of all the digits in all strings in nums_list.

    Precondition: s.isdigit() holds for each string s in nums_list.

    >>> digital_sum(['64', '128', '256'])
    34
    >>> digital_sum(['12', '3'])
    [                                   ]

    >>> digital_sum( [                                   ] )
    [                                   ]

    >>> digital_sum( [                                   ] )
    [                                   ]
```

4. Complete the examples in the docstring and then the function body.

```
def can_pay_with_two_coins(denoms: List[int], amount: int) -> bool:
    """Return True if and only if it is possible to form amount, a number
    of cents, using two coins of any of the denominations in denoms.

    Pre: each element of denoms is positive
         amount >= 0

    >>> can_pay_with_two_coins([1, 5, 10, 25], 35)
    True
    >>> can_pay_with_two_coins([1, 5, 10, 25], 20)
    True
    >>> can_pay_with_two_coins([1, 5, 10, 25], 12)
    [                                  ]

    >>> can_pay_with_two_coins( [                                  ] )
    [                                  ]

    >>> can_pay_with_two_coins( [                                  ] )
    [                                  ]

    >>> can_pay_with_two_coins( [                                  ] )
    [                                  ]

    >>> can_pay_with_two_coins( [                                  ] )
    [                                  ]
```