

CSCI 145 Quiz 4 practice questions on string loops, accumulators, and lists

1. Suppose `nums = [5, 8, 2, 9]`. Write the value of each expression.

<code>nums[1]</code>	_____	<code>nums[-1]</code>	_____
<code>nums[1:3]</code>	_____	<code>len(nums)</code>	_____
<code>2 in nums</code>	_____	<code>nums + [1]</code>	_____

2. Consider this code:

```
pets = ["cat", "dog"]
pets.append("fish")
pets.insert(0, "bird")
x = pets.pop()
pets[1] = "cow"
```

- a) What is `pets` at the end? _____
- b) What is `x` at the end? _____

3. Suppose `nums = [1, 2, 3]` and `word = "cat"`. Which line causes an error? Circle one.

- | | |
|-------------------------------|---------------------------------------|
| A. <code>nums[0] = 7</code> | B. <code>nums.append("j")</code> |
| C. <code>word[0] = "b"</code> | D. <code>word = "b" + word[1:]</code> |

4. The user types 19 when this code runs. What happens? Circle one.

```
age = input("Age? ")
print(age + 1)
```

- | | | | |
|-------------|-----------------|------------------|-------------------|
| A. An error | B. It prints 20 | C. It prints 191 | D. It prints 19 1 |
|-------------|-----------------|------------------|-------------------|

5. Trace `mystery("data")`. Fill in one row per pass through the loop, then write what it returns.

```
def mystery(word):
    result = ""
    for ch in word:
        if ch != "a":
            result = ch + result
    return result
```

ch	result
<i>before loop</i>	

Returns: _____

6. `all_positive(nums)` should return `True` if every number in `nums` is greater than 0. Otherwise, it should return `False`.

```
def all_positive(nums):
    for num in nums:
        if num > 0:
            return True
        else:
            return False
```

- a) What does `all_positive([4, -2, 7])` return?

- b) Rewrite the function so it is correct.

7. Write the exact output of each snippet, one printed line per line.

A

```
nums = [3, 4, 5]
for num in nums:
    total = 0
    total = total + num
print(total)
```

B

```
for i in range(3):
    row = ""
    for j in range(i + 1):
        row = row + str(j)
    print(row)
```

C

```
word = "hello"
for i in range(len(word)):
    if i % 2 == 0:
        print(word[i])
```

D

```
words = ["hi", "cat", "a"]
sizes = []
for w in words:
    sizes.append(len(w))
print(sizes)
```

8. Write a function `above_average(nums)` that returns a new list of the numbers in `nums` that are greater than the average of `nums`, in their original order. If `nums` is empty, it returns `[]`. Otherwise, it returns the new list, which may be empty. For example:

```
>>> above_average([2, 9, 4, 7])
[9, 7]
>>> above_average([5, 5])
[]
>>> above_average([])
[]
```

9. Write a function `index_of_max(nums)` that returns the index of the largest number in `nums`, without using `max()`. Assume `nums` is not empty. If one number is larger than all the others, it returns that number's index. Otherwise, it returns the index of the first of the tied largest numbers. For example:

```
>>> index_of_max([3, 8, 1])
1
>>> index_of_max([3, 8, 1, 8])
1
>>> index_of_max([-4, -2, -9])
1
```