DS2000 - Fall 2024 - Practice Exam #2

This practice exam contains some sample questions on the same topics that will appear on Mini-Exam #2. Please complete the questions on your own, and we'll publish the solutions on Piazza

Mini-Exam #2 takes place on October 29th 2024. It will be administered on paper, during our usual lecture time.

The exam is designed to be approximately 30 minutes long, but you may use the entire class time to complete it. Don't rush, take your time with each question, and double-check your solutions before handing it in.

For the exam, you may bring one 8.5x11-inch piece of paper, with anything written or typed on it (one side only). You may not share your cheat sheet with others. You will submit this cheat sheet along with your exam, and you will not be permitted to use any other materials or notes during the exam.

Part One -- Lists

1A Which of the statements below are reasons we might iterate over a list by position, rather than by value? Select all that apply.

(A) To modify the values in a list.

(B) We never need to iterate by position.

(C) To make sure we don't modify the values in a list.

(D) To iterate over two related lists at the same time.

1B What does the variable **`lst`** contain after the following code runs?

```
lst = [10, 20, 30]
for i in range(1, len(lst)):
   lst[i] = lst[i - 1]
```

Solution:
[10, 10, 10]

1C Write a snippet of Python code that modifies the values in the list below by multiplying all of them by 10**.** Do not create a new list; the original list should change. No need to include a def main(), comments, or anything similar -- just the snippet requested.

```
int_lst = [4, 5, 10, 11, 13, 5, 18]
```

Solution:

```
for i in range(len(int_lst)):
      int_lst[i] = int_lst[i] * 10
```

Part Two -- Functions

2A Which of the following Python snippets is best for calling a function named **`foo`** that takes no parameters and returns an int? Select only one option (the best answer) below.

(A) `foo = ()`

(B) `foo()`

(C) `foo = def()`

(D) `result = foo()`

2B What will the following Python code print?

```
def func(p1):
      return p1 * 2
def main():
      x = 4
      func(x)
      print(x * 2)
      print(func(x * 2))
main()
```

Solution:
8
16

2C Complete the Python function below. Its two parameters are a list of integers, *lst*, and an integer, *k*. Create and return a new list, which is similar to *lst*, except it contains only those values that are greater than *k*. For example, given `[1, 2, 3, 4, 5], 3`, your function should return `[4, 5].` Comments are not required.

```
def big_vals(lst, k):


```

Solution:

```
def keep_big(lst, k):
      new_lst = []
      for num in lst:
            if num > k:
                  new_lst.append(num)
      return new_lst
```

Part Three -- 2D Lists

3A For the 2D list `lst = [[2, 4, 6, 8], [1, 2, 3, 4]]`, circle **all** the expressions below that produce no error and access a *one-dimensional list.*

(A) `lst`

(B) `lst[0]`

(C) `lst[1]`

(D) `lst[1][1]`

(D) `lst[2]`

3B What would the code below print to the terminal?

```
lst = [[2, 4, 6, 8], [1, 2, 3, 4]]
for i in range(len(lst)):
      print(lst[i][1])
```

Solution:
4
2

3C Complete the Python function below. Its two parameters are 2-dimensional list of strings, *names,* and a string *s*. Return an integer, the number of times *s* appears anywhere in *names*. For example, given `[["carol", "grizz"], ["carol", "choux"]], "carol"`, your function should return `2`. (*Hint:* `names.count("carol")` won't give the right answer!) Comments are not required.

```
def count_string(names, s):


```

Solution:

```
def count_string(names, s):
      count = 0
      for row in names:
            count += row.count(s)
      return count
```