**Computer Science 134 – Spring 2020** **Anonymous ID:**

*Iris Howley & Shikha Singh*

Homework 3 – *Due: Monday, March 2 (in class)*

1. In the following table, the value `alice` is used to produce the value of `bob` using an assignment making use of indexing. Fill in the blanks.

| | value of `alice` | value of `bob` | Assignment |
|---|---|---|---|
| e.g. | `"Hello, world"` | `"Hello"` | `bob = alice[0:5]` |
| a. | `"Pixel"` | `"P"` | `bob =` |
| b. | `"February"` | `"bru"` | `bob =` |
| c. | `"Ephraim"` | | `bob = alice[-3:]` |
| d. | `"grace hopper"` | | `bob = alice[6:30]` |
| e. | `"ornation"` | `"onto"` | `bob =` |
| g. | `"desserts"` | `"stressed"` | `bob =` |
| h. | `"blueness"` | `"snub"` | `bob =` |
| i. | | `"tapia"` | `bob = alice[:3]+alice[3:]` |

For the next few questions, we will think about the implications of working with mutable and immutable objects in Python. Beside each `# prints:`, indicate what is printed.

2a.
```
hopper = [ 20, 21, 22 ]  # some upcoming years
tapia =   [ 20, 21, 22 ]  # some classes of students
print(hopper is tapia)   # prints:
hopper.append(23)       ###
print(hopper)          # prints:
print(tapia)            # prints:
```

Explain what is happening to `hopper` and `tapia` (if anything) on the statement marked `###`.

2b.
```
hopper = tapia = [ 23, 20, 21, 22 ] # some upcoming graduating years
print(hopper is tapia)   # prints:
sorted(hopper)          ###
print(hopper)           # prints:
print(tapia)             # prints:
tapia.sort()             ###
print(hopper)           # prints:
print(tapia)             # prints:
```
Explain what is happening on the statements marked ###.

2c.
```
hopper = tapia = 'upcoming years'
print(hopper is tapia)   # prints:
tapia.replace('upcoming', 'graduating') ###
print(tapia)             # prints:
print(hopper)           # prints:
hopper = hopper.replace('upcoming','pre-alumni') ###
print(hopper)           # prints:
print(tapia)             # prints:
```
Explain what is happening on the statements marked ###.

2d.
```
nestedList = [[1, 2], [3, 4]]      # list of lists
nestedList.append(nestedList[1])  ###
nestedList[2][1] = 6              ###
print(nestedList)                 # prints:
```
Explain what is happening on the statements marked ###.

⋆