Sequences III and Mutability

Name:

1. Consider this program:

```
def mystA(b : str) -> str:
    print(b)
    b = 'azul'
    return b

def mystB(a : str) -> None:
    print(a)
    a = 'plav'
    print(a)

def main() -> None:
    a = 'blue'
    b = 'ivy'
    a = mystA(b)
    print(a)
    print(b)
    mystB(a)
    print(a)
    print(b)

main()
```

What is the output of this program? Draw the function frame diagrams (showing the variables in each function frame) while tracing the code.

main's Frame	mystA's Frame	mystB's Frame
a -> ~~'blue'~~ 'azul' a -> ~~'ivy'~~	b -> ~~'ivy'~~ 'azul'	a -> ~~'blue'~~ 'plav'

```
ivy
azul
ivy
azul
plav
azul
ivy
```

(This page is left blank intentionally)

2. Evaluate the following Python expressions in the order that they are listed:

Python code	Output
`s = 'abc'`	
`lst = list(s)`	
`lst[0] = s`	
`print(lst)`	['abc','b','c']
`del lst[0]`	
`print(lst)`	['b','c']
`del s[0]`	Runtime error: str is immutable
`lst += ['a','b','c']`	
`print(lst)`	['b','c','a','b','c']
`del lst[1::2]`	
`print(lst)`	['b','a','c']
`lstc = ['b','a','c']`	
`lst == lstc`	True
`lst is lstc`	False
`id(lst) == id(lstc)`	False
`lsta = lst`	
`lst == lsta`	True
`lst is lsta`	True
`id(lst) == id(lsta)`	True
`lstb = lst[:]`	
`lst == lstb`	True
`lst is lstb`	False
`id(lst) == id(lstb)`	False
`lst.append('d')`	False
`lst += 'e'`	False
`print(lst)`	['b','a','c','d','e']
`lst += 1`	TypeError: int is not a sequence type
`lst == lsta`	True

3. What is the output of the following code snippet?

```
listA = [2, 4, 6]
print(listA)
listA[1] = 40
print("A", listA)
listB = []
print("A", listA, "B", listB)
listB.append(2)
listB.append(40)
listB.append(6)
print("A", listA, "B", listB)
```

```
[2, 4, 6]
A [2, 40, 6]
A [2, 40, 6] B []
A [2, 40, 6] B [2, 40, 6]
```

4. What is the output of the following code snippet?

```
listC = listA
print("A", listA, "C", listC)
listC[0] = 20
print("A", listA, "C", listC)
listA[2] = 60
print("A", listA, "C", listC)
listD = [20, 40, 60]
print("A", listA, "C", listC, "D", listD)
print("A is C?", listA is listC, "A is D?", listA is listD)
print("A == C?", listA == listC, "A == D?", listA == listD)
listE = listA[:]
print("A", listA, "E", listE)
listE[0] = 10
print("A", listA, "E", listE)
listA[2] = 70
print("A", listA, "E", listE)
```

```
A [2, 40, 6] C [2, 40, 6]
A [20, 40, 6] C [20, 40, 6]
A [20, 40, 60] C [20, 40, 60]
A [20, 40, 60] C [20, 40, 60] D [20, 40, 60]
A is C? True A is D? False
A == C? True A == D? True
A [20, 40, 60] E [20, 40, 60]
A [20, 40, 60] E [10, 40, 60]
A [20, 40, 70] E [10, 40, 60]
```

5. What is the output of the following code snippet?

```
def funcX(listR):
    print("R", listR)
    listR[0] = 2
    print("R", listR)

listQ = [1, 3, 5]
print("Q", listQ)
funcX(listQ)
print("Q", listQ)
```

```
Q [1, 3, 5]
R [1, 3, 5]
R [2, 3, 5]
Q [2, 3, 5]
```

6. What is the output of the following code snippet?

```
def swapZero(d, e):
    f = d
    d[0] = e[0]
    e[0] = f[0]

u = [1, 3, 5]
v = [2, 4, 6]
swapZero(u, v)
print("U", u, "V", v)
```

```
U [2, 3, 5] V [2, 4, 6]
```

7. What is the output of the following code snippet?

```
def swapOne(g, h):
    i = g[:]
    g[1] = h[1]
    h[1] = i[1]

w = [1, 3, 5]
x = [2, 4, 6]
swapOne(w, x)
print("W", w, "X", x)
```

```
W [1, 4, 5] X [2, 3, 6]
```

8. What is the output of the following code snippet?

```
def swapTwo(p, q):
    r = p[:]
    s = q[:]
    r[1] = s[1]
    s[1] = p[1]

y = [1, 3, 5]
z = [2, 4, 6]
swapTwo(y, z)
print("Y", y, "Z", z)
```

```
Y [1, 3, 5] Z [2, 4, 6]
```

Trace the code below. For each line of code, write what it prints if it's a print statement or indicate the type of the relationship between variables by indicating alias or copy for statements that are not print statements. It might be helpful to draw diagrams that keep track of the values that different variables point to.

```
def func(x, y):
    y[0] = x[0] + y[0]
    x = x[:]
    print(x)            # Point 1
    print(y)
    y = x

a=[1,2,3]
b=a.copy()
func(a,b)
print(a)                # Point 2
print(b)
```

9. Write down what the program prints here:

```
[1,2,3]
[2,2,3]
[1,2,3]
[2,2,3]
```