Sequences, Mutability, Loops, & Conditionals Practice

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

```
timezones = ['Eastern', 'Pacific']
timezones[0] = ['EDT']
timezones[-1] = 'PDT'
print(timezones)
timezones[1:1] = ['Central', 'Mountain']
print(timezones)
timezones[0:1] = ['Eastern']
print(timezones)
```

```
[['EDT'], 'PDT']
[['EDT'], 'Central', 'Mountain', 'PDT']
['Eastern', 'Central', 'Mountain', 'PDT']
```

2. 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]
```

3. 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]
```

4. 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]
```

5. 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]
```

6. 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]
```

7. 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]
```

8. Write a function called `checkout` that prompts the user for the price of an item until the user enters 0 and returns the total cost for all the items.

```
# version 1
def checkout() -> int:
  total = 0
  moreItems = True
  while moreItems:
    price = float(input('Enter price of item (0 when done): '))
    if price != 0:
      total = total + price
    else:
      moreItems = False
  return total
# version 2
def checkout() -> int:
  total = 0
  price = float(input('Enter price of item (0 when done): '))
  while price != 0:
    total = total + price
    price = float(input('Enter price of item (0 when done): '))
  return total
```

9. Write a function called `diff_type` that takes in a list of numbers and prints `+` when the difference between two consecutive elements of the list is positive, `=` when they are equal, and `-`, otherwise.
For example, `diff_type([3,4,1,7])` must output `'+-+'`,`diff_type([12,30,30])` must output `'+='`, `diff_type([32,16,8,4,2,1,0])` must output `'------'`

```
def diff_type(lst: list) -> None:
    for i in range(len(lst)-1):
        if lst[i+1] - lst[i] > 0:
            print('+', end='')
        elif lst[i+1] - lst[i] < 0:
            print('-', end='')
        else:
            print('=', end='')
```

10. Write a function called `remove_false_original` that takes a list of boolean values and removes all False values from the list.

```
def remove_false_original(boolean_list: list) -> None:
    '''Removes all False values from boolean_list'''
    i = 0
    while i < len(boolean_list):
        if not boolean_list[i]:
            del boolean_list[i]
        else:
            i = i + 1
```

11. Write a function called `remove_false_copy` that takes a list of boolean values and returns a copy of the list with all False values removed.

```
def remove_false_copy(boolean_list: list) -> list:
    '''Creates a copy of boolean_list with all False values removed'''
    copy_list = boolean_list[:]
    i = 0
    while i < len(copy_list):
        if not copy_list[i]:
            del copy_list[i]
        else:
            i = i + 1
    return copy_list
```

12. Write a function called `reverse` that takes a list and returns a copy of the list with the order of items reversed. The original list should not be modified.

```
def reverse(my_list: list) -> list:
    reverse_list = []
    for item in my_list:
        reverse_list[0:0] = [item]
    return reverse_list
```

13. Write a function called `make_dense` that takes in a list and outputs a list with the zero values compressed. For example, `make_dense([3,0,0,7,0,0,0])` must return `[3,2,7,3]`, in which the elements at odd indexes represent the number of zeros between the other values (2 because there are 2 zeros between 3 and 7, and 3 because there are 3 zeros after 7).

```
def make_dense(lst: list) -> list:
    dense_lst = []
    zeros = 0
    for num in lst:
        if num != 0:
            if zeros != 0:
                dense_lst += [zeros]
                zeros = 0
            dense_lst += [num]
        else:
            zeros +=1
    if zeros != 0:
        dense_lst += [zeros]
    return dense_lst
```

14. Write a function called `make_sparse` that takes in a dense list and outputs a list with the zero values expanded.

```
def make_sparse(lst: list) -> list:
    sparse_lst = []
    zeros = 0
    for i in range(0, len(lst),2):
        sparse_lst += [lst[i]]
        sparse_lst += [0]*lst[i+1]
    return sparse_lst
```