Quiz 24b: Copying Objects

To begin:

- Download the linked software from Moodle
- Make a copy of this Google Doc so that a member of your pair can edit it.
- Immediately add students' email addresses below so that all members are credited.
- When you have completed the quiz, one member should select "File → Download → PDF Document" and then submit the saved pdf file to git.

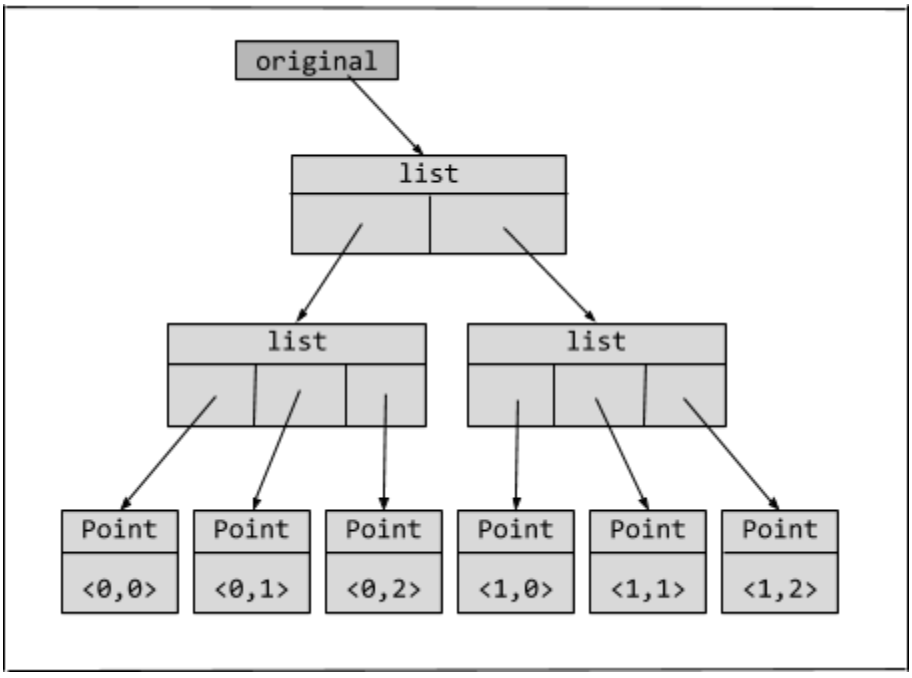
Students:

drew.sadler@slu.edu

For this experiment, we model a two-dimensional table of Point instances formally as a list of lists. For example, the two-dimensional table

<0,0>	<0,1>	<0,2>
<1,0>	<1,1>	<1,2>

will be modeled as a list of two rows, with each row modeled as a list of three points. (Technically, those points really have references to x- and y-coordinates that are their own **`int`** instances, but we choose not to crowd the picture with the portrayal of those explicit **`int`** instances.)



We provide software that builds this model and after doing so, makes a copy of the original model and then lets you experiment by subsequently changing the original model, either by adding an additional row, adding an additional column, replacing an existing Point instance by a new Point instance with chosen coordinates, or by mutating the values of an existing Point instance.

If all goes well, changes to the original model *should not* have a downstream effect on the copies of the original that were made prior to the manipulation. Unfortunately, some of our implementations are flawed. In the first part of this activity, you need not examine the source code but instead run the program and experiment to determine what breaks.

Experiment with each of the four sample behaviors in our experimental software, and place the letter **x** in each entry of the following table cells for which you determine the implementation is **flawed**, namely in which modifying the original matrix has an unwanted side effect of modifying the previously made copy.

algorithm	Mutate point	Overwrite point	Add row	Add column
`copy`	x	x		x
`copyMatrix1`	x	x	x	x
`copyMatrix2`	x	x		x
`copyMatrix3`	x	x		x
`copyMatrix4`	x			
`copyMatrix5`	x			
`copyMatrix6`				
`deepcopy`				

After you have completed the above table, we wish for you to group implementations that seem functionally equivalent in terms of their behavior. That is, there are eight different functions, but there are really only four distinct groups of functions in terms of their behavior.

In the space below, **place all eight of the functions** into four groups based upon those having equivalent semantics. We've started you out by placing four of the eight that truly are distinct from each other. You must place the other four functions within these groups.

`copyMatrix1,`

`copyMatrix3, copyMatrix2, copy`

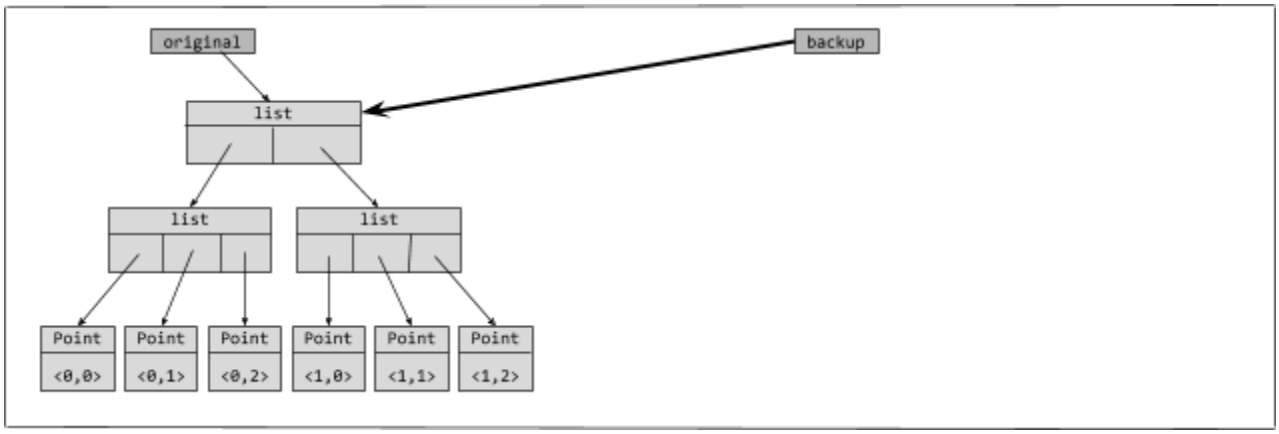
`copyMatrix5, copyMatrix4`

`copyMatrix6, deepcopy`

As the second part of this activity, you are to **examine the source code** in `copyMatrix.py` for the different versions of functions we used to attempt to make a copy. You must demonstrate your understanding of the model and the various objects by illustrating the precise result. Note well that in your illustration, if any part of the result is actually a reference to an EXISTING object from the original model, then you must use an arrow going from the right-side to the left-side to indicate such a reference to that object. Any object newly drawn on the right-hand side will be presumed to be a NEW independent instance.

While this exercise was originally intended to be pen-and-paper, it turns out that Google Docs have a decent interface for creating and editing such pictures. So we are providing you a picture as a starting point together with a label backup that hasn't yet been assigned to anything. For each answer below if you click once on the figure you will see an "Edit" button appear underneath that brings you into the drawing interface. Note well that arrows are within the "Line" menu within. Also, note that if you do wish to make actual copies of parts of our figure, we've grouped things in a natural way that should allow you to copy/paste one of our objects to the right-side of the figure to get your own instance that you can place.

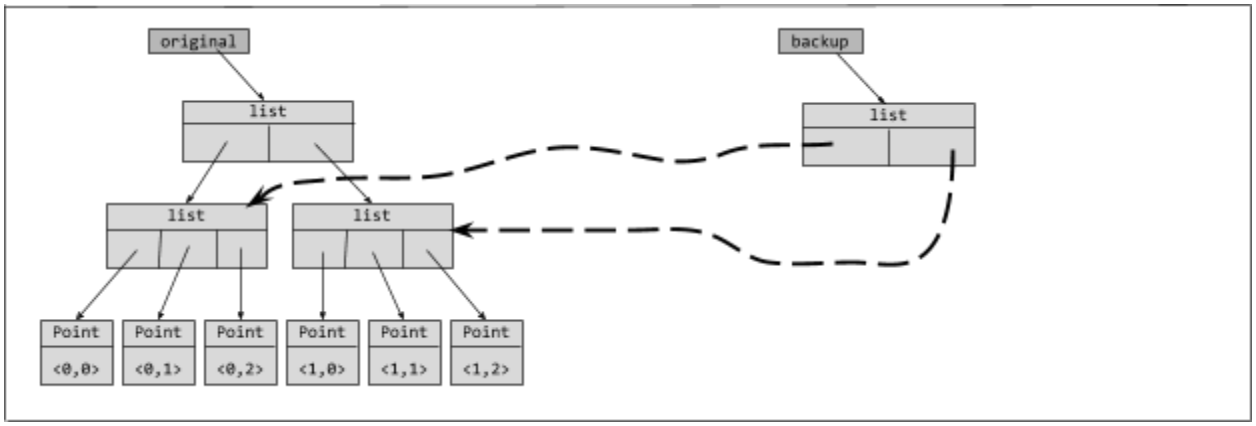
Result of executing `backup = copyMatrix1(original)`



For reference, the implementation of this function appears as follows.

```
def copyMatrix1(m):
    return m
```

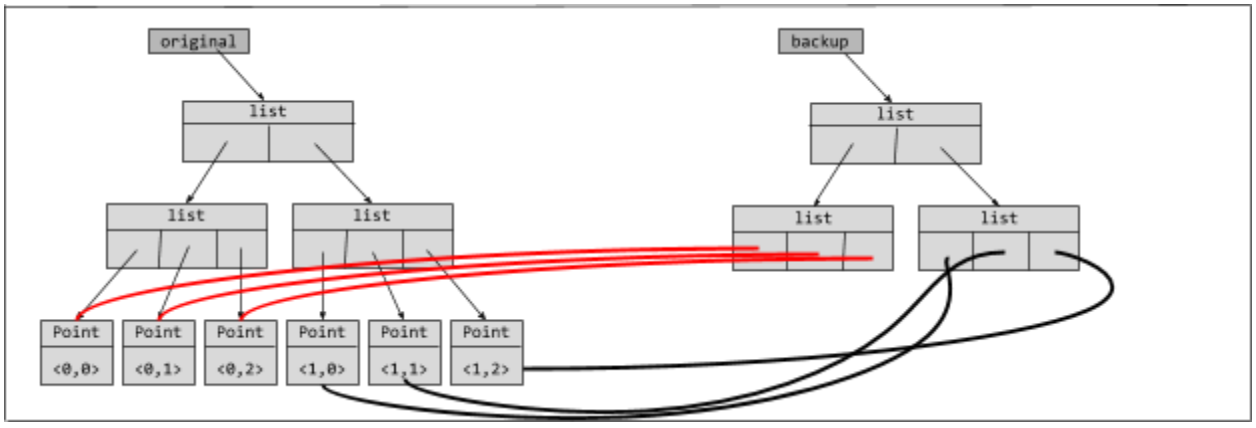
Result of executing `backup = copyMatrix3(original)`



For reference, the implementation of this function appears as follows.

```
def copyMatrix3(m):
    newM = []
    for row in m:
        newM.append(row)
    return newM
```

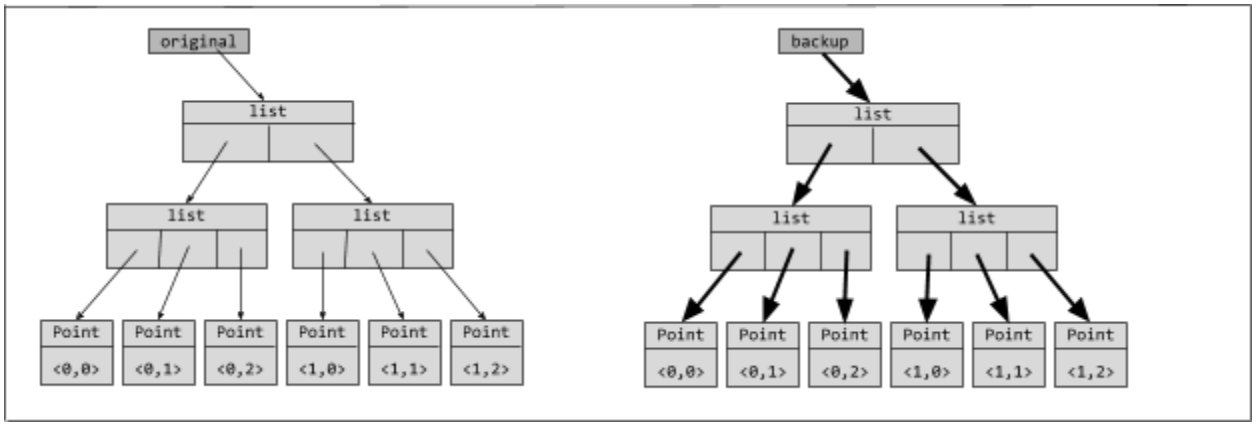
Result of executing `backup = copyMatrix5(original)`



For reference, the implementation of this function appears as follows.

```
def copyMatrix5(m):
    newMat = []
    for row in m:
        newRow = []
        for item in row:
            newRow.append(item)
        newMat.append(newRow)
    return newMat
```

Result of executing `backup = copyMatrix6(original)`



For reference, the implementation of this function appears as follows.

```
def copyMatrix6(m):
    newMat = []
    for row in m:
        newRow = []
        for p in row:
            newRow.append(Point(p.getX(),p.getY()))
        newMat.append(newRow)
    return newMat
```