CS1110 Lab 10: Practice for A7 (May 3-4, 2016)

First Name: _____ **Last Name**: _____ **NetID:** _____

The lab assignments are very important. Remember this: *The lab problems feed into the assignments and the assignments define what the exams are all about.*

Start *before* your lab meets.

We recommend spending an hour or two on the lab *before* coming to your section, so you can use your in-person time to ask questions most efficiently.

Also, this strategy of starting beforehand increases your chances of checking in at your lab section, which will probably take less time than waiting in line at consulting hours!

Getting credit

Complete all required blank boxes and lines on this handout. When you are finished, show your written answers to one of the CS 1110 lab staff in your section on May 3-4 or in any consulting hours up to and including May 9 (earlier days have shorter lines). The staff member will ask you a few questions to make sure you understand the material, and then swipe your Cornell ID card or directly make a notation in CMS to record your success. This physical piece of paper is yours to keep.

Getting set up

Review the Lectures of April 19, 21 and 28. From the Lab webpage, download and unzip `Lab_10.zip` into a folder named (for example) `Lab_10`. In the command shell, navigate the file system so that this folder is THE CURRENT WORKING DIRECTORY.

1 Benchmarking Merge

There are two implementations of merge in `ShowMergeSort.py`: `Merge1` uses `pop` and `Merge2` which does not. The module `BenchMerge.py` is set up for you to do a timing studying that involves these two implementations. What can you say about their relative efficiency as observed for $n = 1000$, 10000, and 100000? What values for `p` and `m` did you use?

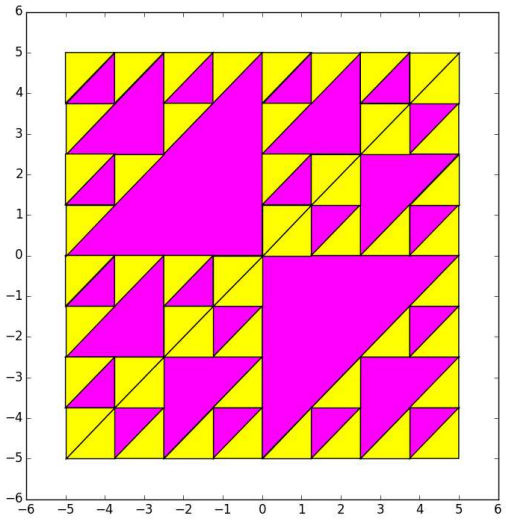
2 Comparing Selection Sort and Merge Sort

The module `BenchSort.py` is set up for you to do a timing studying that compares selection sort and merge sort. What can you say about their relative efficiency as observed for $n = 1000$, 10000, and 100000? What values for `p` and `m` did you use?

3 ShowTriPartition

Play with the module `ShowTriPartition`.
(a) How many yellow triangles are there in a level-L partitioning?

(b) Show how two calls to `Partition` can produce this graphic:



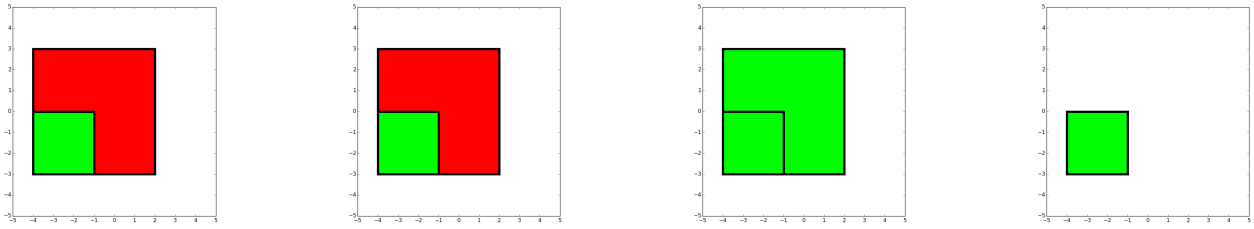
4 Mondrian

Play with the module `ShowMondrian`. For example, add 2 more colors of your own choosing.

Notice that the function `Mondrian` has four recursive calls. Comment out the recursive calls that partition the "northeast" and "southwest" subrectangles. What effect does that change have on the output?

5 Copying Objects

Take a look at the module `ShowSquareClass.py`. When we run its application script we get four figures.



In this exercise you explain the output by drawing state diagrams. The left figure is produced by

```
S1 = Square(-4,-3,6,[1.0,0.0,0.0])
T1 = NestedSquare(S1)
DrawSquare(S1)
DrawSquare(T1)
```

Draw a state diagram that depicts the objects referenced by `S1` and `T1`.

The second from the left figure is produced by

```
S2 = Square(-4,-3,6,[1.0,0.0,0.0])
T2 = deepcopy(S2)
T2.c[1] = 1
T2.c[0] = 0
T2.s = T2.s/2.
DrawSquare(S2)
DrawSquare(T2)
```

Draw a state diagram that depicts the objects referenced by `S2` and `T2`.

The second from the right figure is produced by

```
S3 = Square(-4,-3,6,[1.0,0.0,0.0])
T3 = copy(S3)
T3.c[1] = 1
T3.c[0] = 0
T3.s = T3.s/2.
DrawSquare(S3)
DrawSquare(T3)
```

Draw a state diagram that depicts the objects referenced by `S3` and `T3`.

The rightmost figure is produced by

```
S4 = Square(-4,-3,6,[1.0,0.0,0.0])
T4 = S4
T4.c[1] = 1
T4.c[0] = 0
T4.s = T4.s/2.
DrawSquare(S4)
DrawSquare(T4)
```

Draw a state diagram that depicts the objects referenced by `S4` and `T4`.