Solutions to Practice Final Examination #2

Problem 1—Short answer (10 points)

1a) The values for this problem indicate the total number of gifts in each category if you take the words to “The Twelve Days of Christmas” literally. At the end of our true love’s gift-giving spree, the total haul contains:

- 12 Partridges in pear trees
- 22 Turtle doves
- 30 French hens
- 36 Calling birds
- 40 Gold rings
- 42 Geese-a-laying
- 42 Swans-a-swimming
- 40 Maids-a-milking
- 36 Ladies dancing
- 30 Lords-a-leaping
- 40 Pipers piping
- 22 Drummers drumming

Charles M. Schulz offered a lovely rendition of this problem in 1963:

```
WELL, IT TOOK ME SIX WEEKS, BUT I FINALLY FIGURED IT OUT...

ALL IN ALL, HE GAVE HER:
TWENTY-TWO TURTLE DOVES,
THIRTY-FIVE HENS THIRTY-SIX,
CALLING BIRDS, FORTY-GOLD RINGS,
FORTY-TWO GEASE-A-LAYING...
```

1b) 42. As on the first practice final, the key to answering this question is understanding the scope of the different variable names. The call to the MyClass constructor in the code for example assigns a MyClass object to value in which the variable x in the myFunction attribute is bound to 14. Calling test(8) calls myFunction(8 + 6), which assigns 14 to the parameter y. The result is then 2 * 14 + 14, or 42.
Problem 2—Simple graphics (10 points)

```python
# Constants
TEXT_BOX_MARGIN = 10

def createTextBox(lines, font):
    """
    Creates a GCompound consisting of a rectangular frame that contains
    a set of GLabel objects, one for each line in the string array lines.
    """
    labels = [ ]
    width = 0
    for line in lines:
        label = GLabel(line)
        label.setFont(font)
        if (label.getWidth() > width):
            width = label.getWidth()
        labels.append(label)
    width += 2 * TEXT_BOX_MARGIN
    height = 2 * TEXT_BOX_MARGIN + len(labels) * labels[0].getHeight()
    frame = GRect(0, 0, width, height)
    textBox = GCompound()
    textBox.add(frame)
    y = TEXT_BOX_MARGIN + labels[0].getAscent()
    for label in labels:
        textBox.add(label, TEXT_BOX_MARGIN, y)
        y += label.getHeight()
    return textBox
```
Problem 3—Interactive graphics (15 points)

```python
# This function simulates the Whac-A-Mole arcade game.
def WhacAMole():
    def clickAction(e):
        nonlocal circle
        obj = gw.getElementAt(e.getX(), e.getY())
        if obj is not None and obj.isFilled():
            gw.remove(obj)
            circle = None
    def step():
        nonlocal circle
        if circle is not None:
            circle.setFilled(False)
            x = random.randint(0, GWINDOW_WIDTH)
            y = random.randint(0, GWINDOW_HEIGHT)
            circle = gw.getElementAt(x, y)
            if circle is not None and not circle.isFilled():
                circle.setFilled(True)

gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT)
circle = None
gw.addEventListener("click", clickAction)
timer = gw.createTimer(step, TIME_STEP)
timer.setRepeats(True)
timer.start()

# Creates the pattern of circular holes on the graphics window. The
circles form a matrix whose dimensions are given by N_ROWS and N_COLUMNS.
The diameter of each circle is given by the constant CIRCLE_SIZE and the
space between each circle is given by the constant CIRCLE_SEP.
def createCircles(gw):
    delta = CIRCLE_SIZE + CIRCLE_SEP
    for i in range(N_ROWS):
        y = CIRCLE_SEP / 2 + i * delta
        for j in range(N_COLUMNS):
            x = CIRCLE_SEP / 2 + j * delta
            hole = GOval(CIRCLE_SIZE, CIRCLE_SIZE)
            gw.add(hole, x, y)

# Startup code
if __name__ == "__main__":
    WhacAMole()
```
Problem 4—Strings (10 points)

```python
# File: CreateTocEntry.py

""
This program solves the createTocEntry problem from the practice final.
""

# Constants

TOC_LINE_LENGTH = 60

def createTocEntry(title, page):
    ""
    Creates a table of contents entry in which the chapter title and the
    page number are separated by a leader composed of alternating spaces
    and periods.
    ""
    entry = title + " "
    if len(title) % 2 == 0:
        entry += " "
    pageString = " " + str(page)
    gap = TOC_LINE_LENGTH - len(entry) - len(pageString)
    for i in range(gap):
        if i % 2 == 0:
            entry += "."
        else:
            entry += " "
    entry += pageString
    return entry
```

Problem 5—Arrays (10 points)

```python
# File: FindFirstDuplicate.py

""
This file implements the findFirstDuplicate function from the practice final.
""

def findFirstDuplicate(array):
    ""
    Finds and returns the first element in the array that appears
    more than once in the array. If no duplicated element exists,
    findFirstDuplicate should return the value None.
    ""
    for i in range(len(array)):
        element = array[i]
        if element in array[i + 1:]:
            return element
    return None
```
Problem 6—Recursive functions (10 points)

```python
# File: HailstoneNumber.py

"""
This program finds the hailstone number for an integer using recursion.
"""

# Returns the hailstone number of n, which is the number of steps
# required to reach 1 by executing the following steps repeatedly:
# # - If n is 1, stop.
# # - If n is even, replace n by n / 2.
# # - If n is odd, replace n by 3 * n + 1.

def hailstoneNumber(n):
    if n == 1:
        return 0
    elif n % 2 == 0:
        return 1 + hailstoneNumber(n / 2)
    else:
        return 1 + hailstoneNumber(3 * n + 1)
```

Problem 7—Defining classes (10 points)

```python
# File: PixelArray.py

"""
This module defines the PixelArray class, which implements a two-dimensional
array of integers.
"""

class PixelArray(list):
    def __init__(self, width, height):
        list.__init__(self)
        for i in range(height):
            self.append([0] * width)
```

Problem 8—Linked structures (10 points)

```python
def height(t):
    """Returns the height of the tree t."""
    if t is None:
        return -1
    else:
        return max(height(t._left), height(t._right)) + 1
```
Problem 9—Python data structures (15 points)

```python
# File: AdventureCheatSheet.py

""
This file defines the AdventureCheatSheet function, which prints a list
of all Adventure objects and their uses.
""

from AdvGame import AdvGame

def printCheatSheetForObjects(objects, rooms):
    for obj in objects:
        obj = objects[obj]
        desc = obj.getDescription()
        loc = obj.getInitialLocation()
        if loc != "PLAYER":
            loc = rooms[loc].getShortDescription()
        print(objName + " " + desc + ": " + loc)
        for room in rooms:
            room = rooms[roomName]
            text = room.getText()
            if key == objName:
                short = room.getShortDescription()
                print(" Needed for " + dir + " from " + short)
```