2) Consider the following code:

```python
import random
tots = [0.0]*3
maxVals = [0.0]*3
mean = 100.0
stdDevs = [0.0, 20.0, 40.0]
for i in range(1000):
    for j in range(len(tots)):
        next = random.gauss(mean, stdDevs[j])
        tots[j] += next
        if next > maxVals[j]:
            maxVals[j] = next
```

2.1. What are the expected values of each element of `tots` at the end of the code? Hint: `random.gauss(mu, sigma)` returns a random value chosen from a Gaussian distribution with mean `mu` and standard deviation `sigma`. (9 points)

100,000 100,000 1000,000

Is each of the following True or False (9 points):

- 2.2. One would expect `maxVals[0]` to be less than `maxVals[1]`. **TRUE**
- 2.3. One would expect `maxVals[1]` to be less than `maxVals[2]`. **TRUE**
- 2.3. If the code were run twice, the value of `tots[0]` would be the same each time. **TRUE**
3) The code produces 4 plots. Match each plot below to a figure number by writing 1, 2, 3 or 4 on the plots. (16 points)

```python
y1, y2, y3, y4 = [], [], [], []
for i in range(100):
    y1.append((i**2)/50.0)
    y2.append(2*i)
for i in range(99):
    y3.append(y1[i+1] - y1[i])
    y4.append(y2[i+1] - y2[i])
pylab.figure(1)
pylab.plot(y1)
pylab.figure(2)
pylab.plot(y2)
pylab.figure(3)
pylab.plot(y3)
pylab.figure(4)
pylab.plot(y4)
```

1

2

3

4
5) Write a function that uses a Monte Carlo simulation to find the probability of a run of at least 4 consecutive heads out of ten flips of a fair coin, and then returns that probability. Assume that 10,000 trials are sufficient to provide an accurate answer. You may invoke the function:

```python
def simThrows(numFlips):
    """Simulates a sequence of numFlips coin flips, and returns True if the sequence contains a run of at least four consecutive heads and False otherwise."""

def sim():  # write your code below
    return runSim(100000, 4)
```

```python
def runSim(numTrials, numThrows):
    numSuccess = 0.0
    for t in range(numTrials):
        if simThrows(numThrows):
            numSuccess += 1.0
    return numSuccess/numTrials

if __name__ == '__main__':
    print(sim())
```