2) Consider the following code:

```python
def oneTest():
    tries = 0
    while True:
        tries += 1
        ind1 = random.choice(range(52))
        ind2 = random.choice(range(52))
        if ind1 == ind2: break
    return tries

def makePlots(numTrials, oneTest):
    numTries = []
    for t in range(numTrials):
        numTries.append(oneTest())
    pylab.plot(numTries)
    pylab.figure()
    pylab.hist(numTries, bins = 10)
makePlots(1000, oneTest)
```

2.1. Write Python code that calculates the width of each bar (they are all the same width) in the histogram. (5 points)

```python
minVal, maxVal = pylab.xlim()
width = (maxVal - minVal) / 10.0
```

2.2. Assuming that the width of each bar is $w$, describe the range of values on the x axis covered by the tallest bar in the histogram. (5 points)

$(\text{minVal}, \text{minVal} + w)$

2.3. Is it likely that the call to `pylab.plot` would produce a plot similar to one below? (5 points)

Yes.
John had a strategy for eventually winning a lottery with 1000 tickets. The first time he entered he would buy one ticket. If he didn’t win he would double the number of tickets (to two) he bought the next time he entered. If he didn’t win that time, he would double the number of tickets (to four) again. Etc. What is the probability that John wins the lottery before playing it 4 times? (10 points)

The chance of John winning the first time is 1/1000 (assuming the tickets are uniformly distributed).

If he doesn’t win, he enters the second lottery. The probability of his not winning the first is 999/1000. The probability of his winning the second lottery is 2/1000.

Applying the same argument for the third lottery we have the probability of John winning the lottery before playing it 4 times is:
\[
\frac{1}{1000} + \frac{999}{1000} \times \frac{2}{1000} + \frac{999}{1000} \times \frac{998}{1000} \times 4/1000
\]

Alternately, one can compute the probability of John not winning the first lottery AND not winning the second AND not winning the third. These are independent events and we can write this probability as: \( \frac{999}{1000} \times \frac{998}{1000} \times \frac{996}{1000} \).

The probability of John winning before playing the lottery 4 times is:
\[
1 - \left( \frac{999}{1000} \times \frac{998}{1000} \times \frac{996}{1000} \right)
\]
which is the same as the previous answer.
4) Write a function that uses a Monte Carlo simulation to estimate the probability of John winning the lottery within \( n \) attempts, assuming he uses the strategy of problem 3. Assume that 10,000 trials are sufficient to provide an accurate answer. You may call the function:

```python
def runLottery(ticketsSold, ticketsBought):
    """ticketsSold is the number of tickets sold in a lottery and ticketsBought is the number of tickets bought by John. It returns 1 if John won the lottery and 0 otherwise."""
```

(20 points)

```python
def sim(n, ticketsSold):
    numWins = 0
    for i in range(10000):
        for j in range(n):
            if runLottery(ticketsSold, 2**j):
                numWins +=1
                break
    return float(numWins) / 10000
```