Problem #1

```python
def genRegions(n):

    # generate n regions and store them as tuples in a list

    regions = []
    for i in range(0, n):
        low = np.random.uniform(-10, 11)
        high = np.random.uniform(-10, 11)
        if (low < high):
            regions.append((low, high))
        else:
            regions.append((high, low))
    return regions

regions = genRegions(10000)
```

```python
# PROBLEM 1

def contains(region, point):

    # check if region contains points

    if (point <= region[1]) and (point >= region[0]):
        return True
    return False
```

Problem #2

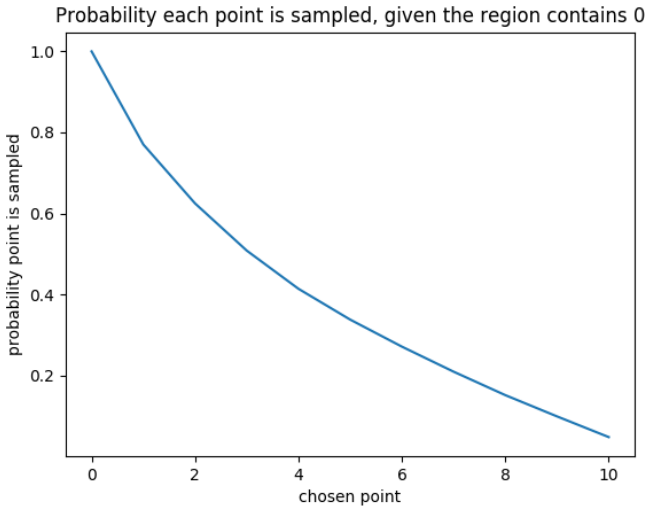
```python
# PROBLEM 2

def findProb(point, regions):

    # finds the probability a region in regions contains point
    # if it contains 0 as well

    weight = 1 / len(regions)

    # find P(x = 0 & x = 1)
    num = 0
    den = 0
    for region in regions:
        height = 1 / (region[1] - region[0])
        if (contains(region, 0)):
            den += weight * height
            if contains(region, point):
                num += weight * height
    p1 = num / den

    # find P(x = 0)
    num = 0
    den = 0
    for region in regions:
        height = 1 / (region[1] - region[0])
        if (contains(region, 0)):
            num += weight * height
            den += weight * height
    p2 = num / den

    return (p1 / p2)
```

Terminal Output:

```
The probability of getting x=1 for regions containing x=0 is: 0.7702903234391875
```

Problem #3

```
# PROBLEM 3

def findProbOverRange(rStart, rEnd, regions):

    # plots the probability for each point in [rStart, rEnd) that it
    # is in a region in regions given that region contains 0

    probabilities = []

    for p in range(rStart, rEnd):
        probabilities.append(findProb(p, regions))

    return probabilities

y = findProbOverRange(0, 11, regions)
x = np.arange(0, 11)
plt.xlabel("chosen point")
plt.ylabel("probability point is sampled")
plt.title("Probability each point is sampled, given the region contains 0")
plt.plot(x, y)
plt.show()
```



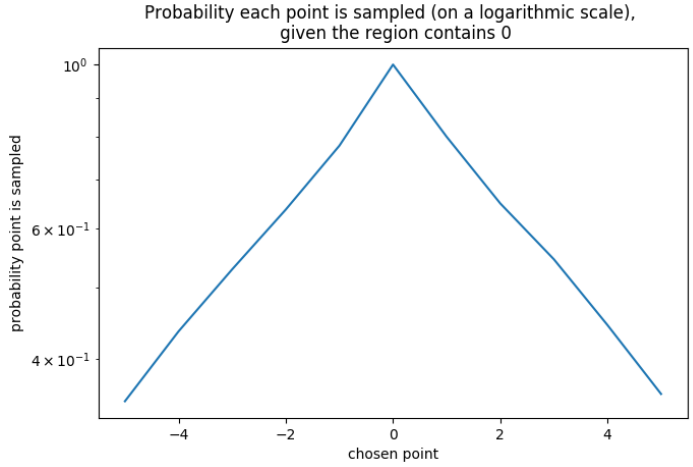
From the graph above, we see this function gives us a downwards-sloping plot, starting with a probability of 0.13 when x = 0, and a probability close to 0.03 when x = 10. Considering that we are testing the probability of sampling 1 from a region that contains 0, this makes a lot of sense. One would expect numbers closer to 0 to have a higher probability of being sampled. For example, for the number 2 to be included in the region, we need our endpoint to be any number in the range [2, 10]. Conversely, for the number 9 to be included in the region, we need our endpoint to be 9 or 10, and that's it. Since we're choosing our starting and ending points randomly, the first scenario is much more likely, and thus the chances of number closer to 0 being included in the region are much higher than numbers further away.

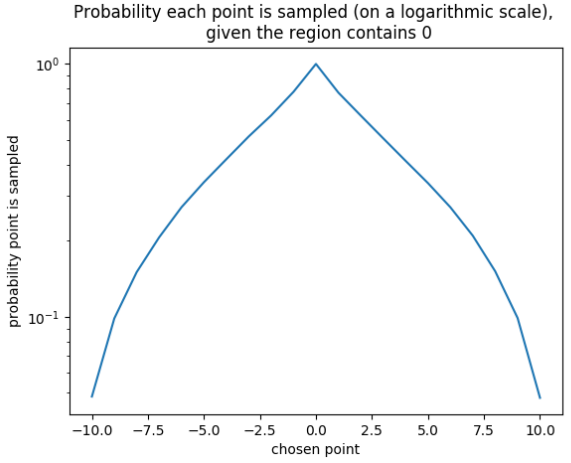
Problem #4

If we were to plot a graph with a linear scale, and the plot had an exponential curve to it, we could get a line that curves and plateaus at a point. When we make the y axis logarithmic, we are now basically testing whether the plot has an exponential decrease; if the plot on a logarithmic axis comes out straight, it means that the thing we are measuring is decreasing / increasing by powers of 10 as we travel along our x axis, indicating the plot is changing by an exponential decrease. A straight plot on a logarithmic scale would look like a curving and then flattening plot on a linear scale.

Problem #5

```
# PROBLEM 5

y = findProbOverRange(-5, 6, regions)
x = np.arange(-5, 6)
plt.xlabel("chosen point")
plt.ylabel("probability point is sampled")
plt.title("Probability each point is sampled (on a logarithmic scale), \n given the region contains 0")
plt.semilogy(x, y)
plt.show()

y = findProbOverRange(-10, 11, regions)
x = np.arange(-10, 11)
plt.xlabel("chosen point")
plt.ylabel("probability point is sampled")
plt.title("Probability each point is sampled (on a logarithmic scale), \n given the region contains 0")
plt.semilogy(x, y)
plt.show()
```

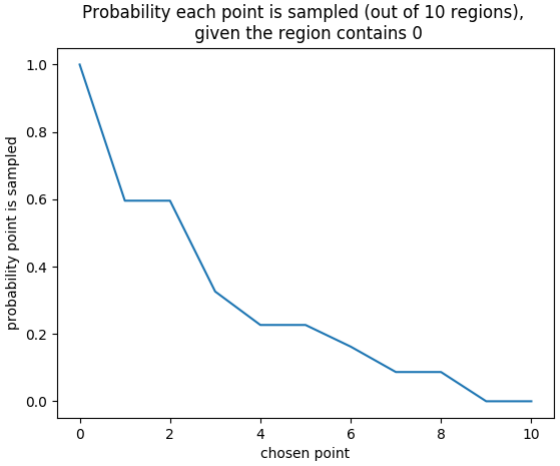


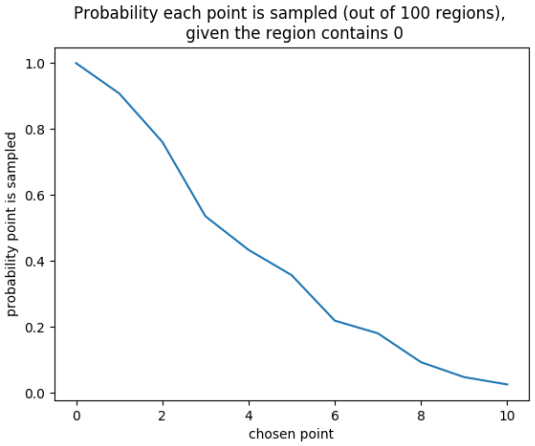


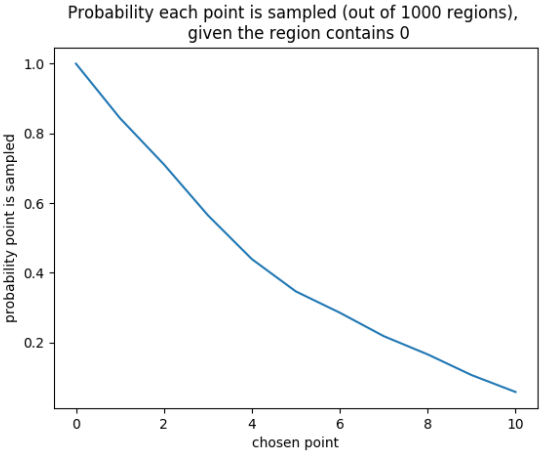
The plots show us that there is an exponential increase in probability for points between -5 and 0, and then an exponential decrease in points 0 through 5. The plots show us that around the central point we still have a similar exponential increase and decrease in probability, but when the plots get closer to the edges, they begin to curve, and start increasing/decreasing at a faster rate This would indicate that the probability when the points get further away from 0 begins to increase / decrease at a rate slower than before (as in, y^2x instead of y^x). This remains in line with our intuition about how, as we travel to points further from zero, our probability that that point is sampled in a region containing 0 gets increasingly lower.

Problem #6

```
# PROBLEM 6
r = int(np.random.uniform(0, 9990))

y = findProbOverRange(0, 11, regions[r : r + 10])
x = np.arange(0, 11)
plt.xlabel("chosen point")
plt.ylabel("probability point is sampled")
plt.title("Probability each point is sampled (out of 10 regions), \n given the region contains 0")
plt.plot(x, y)
plt.show()

r = int(np.random.uniform(0, 9900))

y = findProbOverRange(0, 11, regions[r : r + 100])
x = np.arange(0, 11)
plt.xlabel("chosen point")
plt.ylabel("probability point is sampled")
plt.title("Probability each point is sampled (out of 100 regions), \n given the region contains 0")
plt.plot(x, y)
plt.show()

r = int(np.random.uniform(0, 9000))

y = findProbOverRange(0, 11, regions[r : r + 1000])
x = np.arange(0, 11)
plt.xlabel("chosen point")
plt.ylabel("probability point is sampled")
plt.title("Probability each point is sampled (out of 1000 regions), \n given the region contains 0")
plt.plot(x, y)
plt.show()
```







These graphs show us the evolution of the plot as we increase the number of regions in our sample. We can see that in the graph with only 10 regions, the probabilities are very sporadic and unreliable, but generally hold a similar overall shape. They curve itself becomes much smoother the more trials we perform.

Problem #7

In order to determine how many regions someone used in their generalization, we would need to plot graphs of probabilities over ranges of points and see how jagged the graph is. On our own end, we can plot multiple graphs with varying region numbers (as we did in problem 6) and try our best to match the graphs in terms of general shape and sporadic-ness. Since our regions are determined randomly, it would be very difficult to automize this process; it would require the human eye to make sound judgements, since we are matching based on overall shape rather than strict numbers.

However, this sort of "eye-balling", guess-and-check process would only work up to a certain point. As we saw in problem 6, the graph starts to become quite smooth when we have 1000 regions. As we increase this number, the plot will only become even smoother – and this will be very difficult, if not impossible, for a human eye to detect. So, we could absolutely tell the difference between 10 and 10,000 regions, but would not be able to tell the difference between 10,000 and 20,000 regions. This makes sense, since the probabilities are supposed to plateau and hit their most accurate value with more and more trials up to a certain point. We only need so many trials in order to get accurate probabilities; the notion of "more accurate" probabilities, that may come from more trials, becomes negligible and unimportant.