In [34]:

```
import numpy as np
```

In [35]:

```
import matplotlib.pyplot as plt
```

In [46]:

```
#1. What is the probability that x=1 is in the concept (e.g. in any consequential region) given that
#x=0
#drawing consequential regions with left edge and right edge(containing 0) totaling 10,000
num=10000
a=-10
b=0
c=10
x=1
def count_prob(num, a,b,c, x):
    count=0
    bad=0
    left_edge=list(np.random.uniform(a,b,num))
    right_edge=list(np.random.uniform(b,c,num))
    end_points=list(zip(left_edge,right_edge))
    for i in range(num):
        if end_points[i][0]<=x and end_points[i][1]>=x:
            count+=1/(end_points[i][1]-end_points[i][0])
        else:
            bad+=1/(end_points[i][1]-end_points[i][0])
    return count/(count+bad)
```

In [37]:

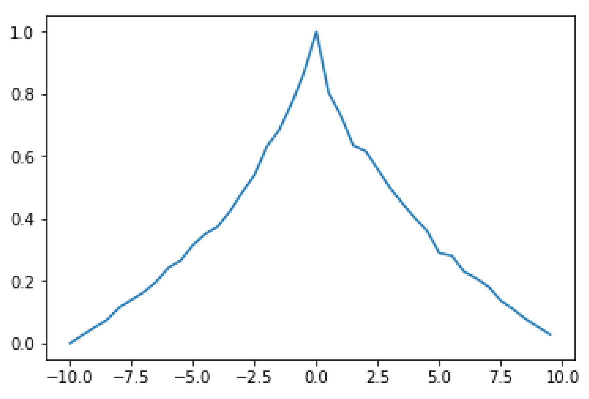
```
count_prob(num, a,b,c, x)
```

Out[37]:

```
0.7508289900390358
```

In [48]:

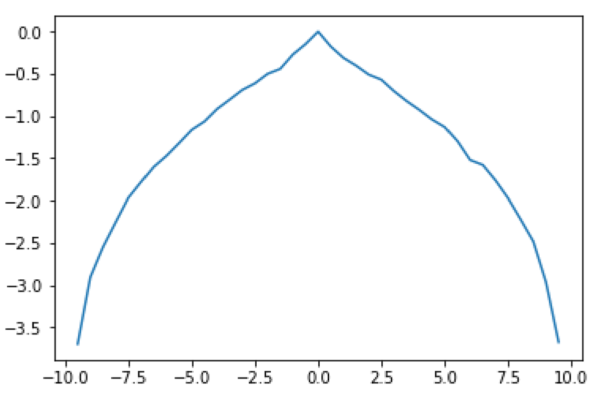
```
#2. Plot the probability that x is in a consequential region as a function x. What does this
#function look like? Write a sentence explaining why intuitively。
step=0.5
y=[]
x_trial=np.arange(a,c,step)
for i in x_trial:
    y.append(count_prob(num,a,b,c,i))
plt.plot(x_trial,y)
plt.show()
#Firstly. there are more consequential regions when approaching the center of the interval, and less consequential regions co
uld be formed approching
#the edges of the interval. The curve at the top of the figure illustrates the gradient of generalization obtained by integra
ting over just these
#consequential regions. The profile of generalization is always concave regard less of what values p(h|x) takes on, as long a
s all hypotheses
#of the same size (in one bracket) take on the same probability according to bayes rules.
```



In [39]:

```
#3. One way to check if the curve has an exponential decrease is to plot a logarithmic y axis and
#look for a straight line. Why does this check if the curve is exponential?
step=0.5
y=[]
x_trial=np.arange(a,c,step)
for i in x_trial:
    y.append(count_prob(num,a,b,c,i))
plt.plot(x_trial,np.log(y))
plt.show()
#if the original curve is exponential and we take log of it, we will get log(e^x)=constant which will give us a straight line
( towards center of interval) that is why we can conclude
#the curve is exponential.
```

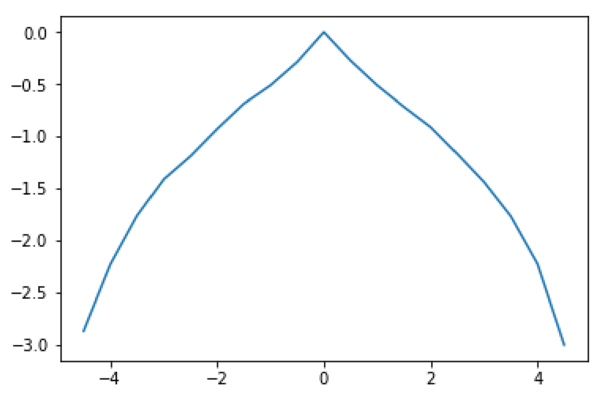
```
/Users/xiaoyingliu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7: RuntimeWarning: divide by zero encounte
red in log
  import sys
```



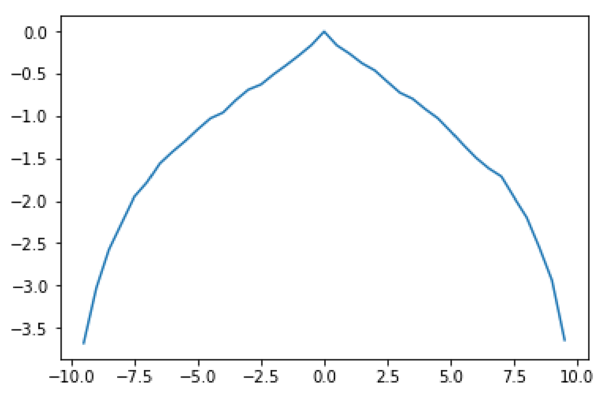
In [44]:

```
#4. Plot a logarithmic y axis for x ranging from -5 to 5, and x ranging from -10 to 10. What do
#these two plots show? How do you interpret them? Explain in a few sentences
#plot of x ranging [-5,5]
a=-5
c=5
step=0.5
y=[]
x_trial=np.arange(a,c,step)
for i in x_trial:
    y.append(count_prob(num,a,b,c,i))
plt.plot(x_trial,np.log(y))
plt.show()


##plot of x ranging [-10,10]
a=-10
c=10
step=0.5
y=[]
x_trial=np.arange(a,c,step)
for i in x_trial:
    y.append(count_prob(num,a,b,c,i))
plt.plot(x_trial,np.log(y))
plt.show()

#the plot with x ranging [-5,5] looks more of a straight line(i.e. the slope is smoother for [-5,5] range compared to [-10,1
0] range. This mean that the
#smaller the interval is, aka, the more centered the interval is, the more it will follow exponential curve, thus the log of
 the curve will look more like
#a straight line.)
```

```
/Users/xiaoyingliu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11: RuntimeWarning: divide by zero encount
ered in log
  # This is added back by InteractiveShellApp.init_path()
```

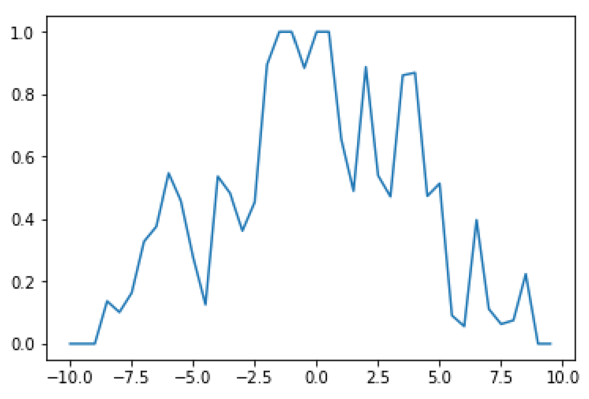


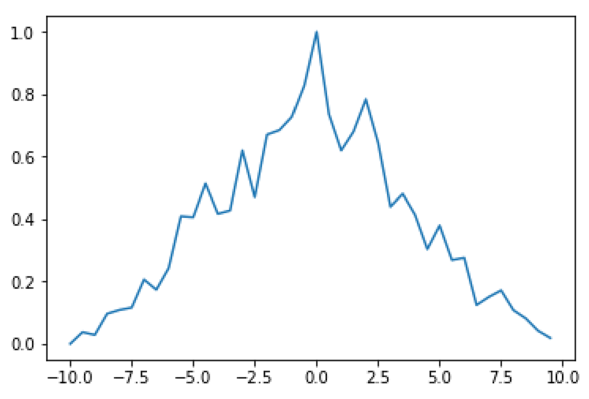

```
/Users/xiaoyingliu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:23: RuntimeWarning: divide by zero encount
ered in log
```

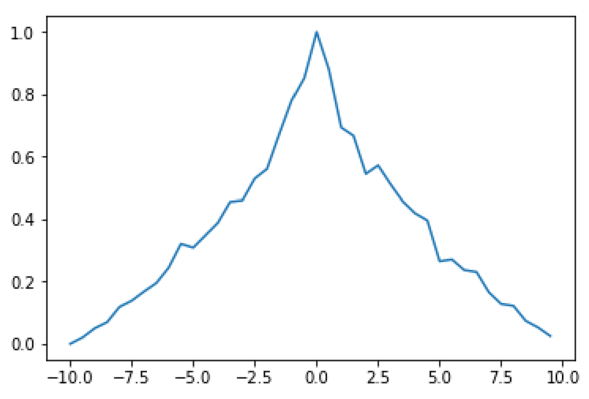


In [31]:

```
#5.In previous questions, we've been assuming that people implement the law perfectly and we
#have been trying to approximate their behavior using 10,000 regions. However, people themselves have
#limited resources. What if people themselves only used a few consequential regions in order to
#compute generalizations? Re-plot Question 2 using only 10, 100, and 1000 consequential regions. What
#patterns do you see?
#replot of 10 regions
a=-10
b=0
c=10
x=1
num=10
step=0.5
y=[]
x_trial=np.arange(a,c,step)
for i in x_trial:
    y.append(count_prob(num,a,b,c,i))
plt.plot(x_trial,y)
plt.show()

#replot of 100 regions
a=-10
b=0
c=10
x=1
num=100
step=0.5
y=[]
x_trial=np.arange(a,c,step)
for i in x_trial:
    y.append(count_prob(num,a,b,c,i))
plt.plot(x_trial,y)
plt.show()

#replot of 1000 regions
a=-10
b=0
c=10
x=1
num=1000
step=0.5
y=[]
x_trial=np.arange(a,c,step)
for i in x_trial:
    y.append(count_prob(num,a,b,c,i))
plt.plot(x_trial,y)
plt.show()
#conclusion:
#when we have limited resources, the curve will not behave as smooth as 10000 trials. the overall shape willl be similar,howe
ver, there are fluctuations
#from the curve. the more trials we can do, the more accurate we will get according to the model assumption(i.e. exponential
 curve)
```







In [ ]:

```
#6. Describe a way you could test how many consequential regions people actually made use of
#in this kind of generalization. Could you tell the difference between 10 and 10,000? Could you tell the
#difference between 10,000 and 20,000, why or why not?

# Answer: considering the accuracy and limitation of resources,  a systematic way for us to test how many consequntial region
s we need will be starting
#from a relatively low number, such as 10. And we look into the pattern generated by the assumption. If there are too many fl
uctuatoins and do not show
#a clear pattern according to bayes rule. Then we will need to increase number of regions to 100. In the meantime, maintaing
 a adjustable, packaged function
#is important for these consecutive trials. I tried the systematic approach myself and find out that 2500-3000 consequential
 regions are usually enough to get
#a smooth curve to recognize pattern.

#10 consequential regions will derive a huge difference from 10000 consequential regions , but 10000 and 20000 consequential
 regions do not make a
#big difference,not that it is easy to observe.
```