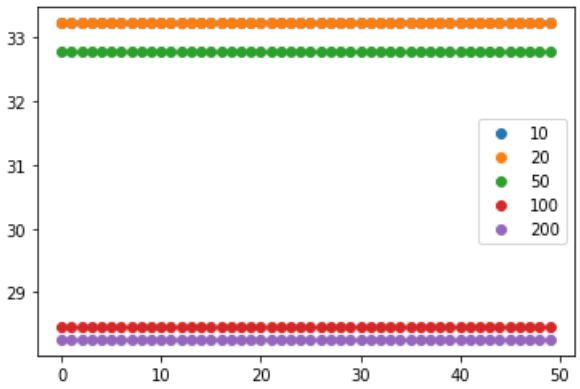
EE325

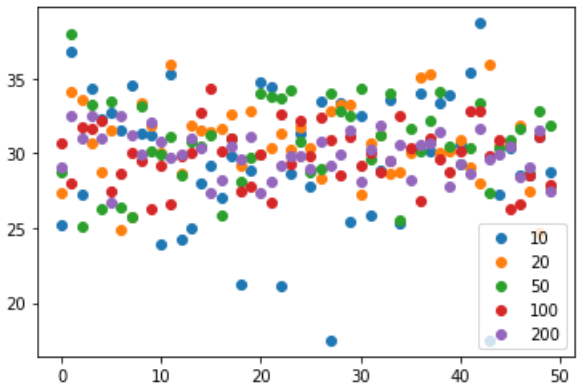
Assignment 1

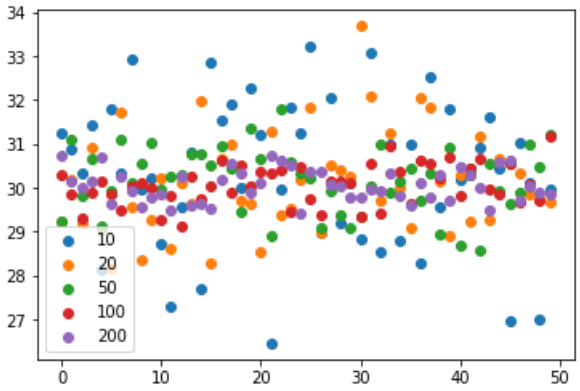
Samar Agarwal, Vasu Soni, Tirth Joshi

In [24]:
```
import random
from matplotlib import pyplot as plt
import numpy as np

data = np.loadtxt("hw1a.txt", dtype=float)



def Average(data, k):
    sum = 0
    for x in range(0, k):
        sum = sum + data[x]
    average = sum/k
    return(average)

def Std(data,x):
    var = 0
    for i in range(0,10000):
        var += (data[i]-x)**2
    std = (float(var)/10000)**(0.5)
    return std

# PART A
def function1(k):
    count = 0
    for f in range(0, k):
        count = count + data[f]
    l = count/k
    sc_A = [l]*50
    plt.scatter(np.arange(50), sc_A)
#function1(10)
#function1(20)
#plt.show()

# PART B
def scenarioB(K):
    sum = 0
    r = random.randint(0, 10000-K)
    for j in range(r, r+K):
        sum = sum + data[j]
    average = float(sum)/K
    return average


def function2(k):
    sc_B = []
    for g in range(0, 50):
        sc_B.append(scenarioB(k))
    plt.scatter(np.arange(0, 50), sc_B)
    return sc_B



# PART C
def function3(k):
    sc_C = []
    for h in range(0, 50):
        data3 = random.sample(sorted(data), k)
        sc_C.append(Average(data3, k))
    plt.scatter(np.arange(0, 50), sc_C)
    return sc_C



def print_graph(Nostudent,i):
    #print(Nostudent)
    if (i==0): function1(int(Nostudent))
    elif(i==1): function2(int(Nostudent))
    elif(i==2): function3(int(Nostudent))
```

```
K = [10, 20, 50, 100, 200]
#t = input("type\n")
for i in range(0,3):
    for t in range(0,5):
        print_graph(K[t],i)
    plt.legend(["10","20","50","100","200"])
    plt.show()
    plt.clf()
Avg_ = Average(data,10000)
Std_ = Std(data,Avg_)
print(Avg_)
print(Std_)

#30.0496
```








```
30.049636297651443
4.935768350155756
<Figure size 432x288 with 0 Axes>
```

What is your guess for the actual average and the actual standard deviation? And what is the actual value?

ANS --> Looking at the data and some of the plots the best guess for the average value of the data is 30.25 and the standard deviation 4 the actual average of the data is 30.0496 and the standard deviation is 4.94

Each of the fifty repetitions can be seen to be a separate survey. If you could do the survey only once for a given K, A. Which of the above three schemes would you use in practice to determine the best guess?

ANS--> Since we don't know the data and we have to the survey only once all the ways of collecting the data
are equally likely but if we are given the data or if we can do the survey more than once the 3rd way of choosing all the students randomly will be the best choice for collecting the data.

B. If you were allowed to choose the value of K, what value would you choose?

ANS--> Since the experiment is done only once and we don't know the data before-hand so any value of K can be taken for us all will be equally likely but if we could repeat the experiment for different values
of K then the best value after looling at the graphs is 100 the data corresponding to this K is very near
to the actual average, but ofcourse 200 is more closer still taking cost into account it is more practicle
value interms of actual data and cost.

B. And how sure would you be of the actual values of the average? What kind of quantitative measure would you use to describe your "sureness of the estimate from the single survey of K samples?"

ANS--> The quantitative measure of sureness can be described by minimizing the errors from actual value.

In [25]:
```
def Output(fname):
    file1 = open(fname, "r")
    data = file1.readlines()
    #print(data)
    print("for the input file "+fname)
    prob = []
    min_prob = 0.425
    rep = 10
    least_prob = 0.355
    rep_ = 20

    heads = 0
    for j in range(0, 100):
        if(data[j] == "1\n"):
            heads += 1

        prob.append(round(float(heads)/(j+1), 3))

    #print (prob)
    check = 0
    check_ = 0
    for x in range(0,86):
        for g in range(0,rep):
            if(prob[x+g]>min_prob):
                break
            else:
                if (g==rep-1):
                    check = 1
                    print("He may begin to doubt at ",x+1+g)
                    break
        if(check == 1):
            break
    for x in range(0,86):
        for g in range(0,rep_):
            if(prob[x+g]>least_prob):
                break
            else:
                if (g==rep_-1):
                    check_ = 1
                    print("His belief must break at ",x+1+g)
                    break
        if(check_ == 1):
            break
    if (check == 0): print("No instance found when he may begin to doubt!!")
    if (check_ == 0 & check == 1): print("No instance found when his belief shall break!!")
```

```
(Output("hw1b1.txt"))
(Output("hw1b2.txt"))
(Output("hw1b3.txt"))
(Output("hw1b4.txt"))
```

```
for the input file hw1b1.txt
No instance found when he may begin to doubt!!
for the input file hw1b2.txt
He may begin to doubt at  14
His belief must break at  36
for the input file hw1b3.txt
No instance found when he may begin to doubt!!
for the input file hw1b4.txt
He may begin to doubt at  10
His belief must break at  20
```

At what point in the sequence does he begin to doubt his initial assumption?

```
ANS--> Kohli will began to doubt his initial assumption if he finds the probability of Heads coming up
staying
    cosequently(say below 0.425 for 10 tosses) below his assumed probability.
```

And when can he be sure that his initial assumption is wrong, if it is indeed wrong?

```
ANS--> He can be sure only when he sees that the probability of Heads coming up staying cosequently(say
below
    0.355 for 20 tosses) which is of course below 0.5 probability of a fair coin.
```

Repeat for three more coins. The results of 100 tosses of these coins are in files hw1b2.txt–hw1b4.txt.

```
ANS--> these results are printed directly in the terminal :)
```

At every point in the sequence, when you are determining whether you are sure of your initial assumption or not, think about how you would describe the "degree of sureness," i.e., provide a quantitative description to your 'hunch.'

ANS--> Degree of sureness can be quantitatively described by checking its error from our guessed probability and 0.5 probability of the coin being fair if the data is close to anyone of these we can be sure for coin being biased on our side or on being fair.

In [26]:
```
import numpy as np

with open(r"hw1c1.txt","r") as file:
    data = file.read()
    data = data.replace('(','')
    data = data.replace(',',' ')
    data = data.replace(')','')
    #print(data)
with open(r"hw1c1.txt","w+") as file:
    file.write(data)
np_arr = np.loadtxt("hw1c1.txt",dtype=float)
A = np.copy(np_arr)
B = np_arr[:,1]
for i in range(0,50):
    A[i][1]=1

TA = A.T
TAA = np.matmul(TA, A)
iTAA = np.linalg.inv(TAA)
TAB = np.matmul(TA, B)
C = np.matmul(iTAA,TAB)
a = C[0]
b = C[1]
#print(a)

file_ = open(r"hw1c2.txt","r")
data_ = file_.read().split("\n")
#print(data_)
#data1 =
```

```
x1=155
x2=165
x3=175

gy1 = a*x1 + b
gy2 = a*x2 + b
gy3 = a*x3 + b
print(C)
sum1 = 0
avg1=avg2=avg3 = 0
for j in range(1,26):
    sum1 = (gy1-float(data_[j]))**2
    avg1 =+ gy1-float(data_[j])
r1 = sum1/25
a1 = avg1/25
std1 = r1**(0.5)
print((100*a1)/gy1)
print((std1*100)/gy1)

sum2  = 0
for j in range(27,52):
    sum2 = (gy2-float(data_[j]))**2
    avg2 =+ gy2-float(data_[j])
r2 = sum2/25
a2 = avg2/25
std2 = r2**(0.5)
print((100*a2)/gy2)
print((std2*100)/gy2)

sum3 = 0
for j in range(53,78):
    sum3 = (gy3-float(data_[j]))**2
    avg3 =+ gy3-float(data_[j])
r3 = sum3/25
a3 = avg3/25
std3 = r3**(0.5)
print((100*a3)/gy3)
print((std3*100)/gy3)

#I guess that avg of the errors might be around +-0.1%
#and std maybe be less than 5% compared to the predicted y value
```

```
[ 0.32447423 -1.09441579]
0.08554119730779093
0.42770598653895464
-0.06838767438080733
0.34193837190403664
-0.4032712259525791
2.0163561297628956
```

Write a program to fit the straight line to this data, i.e., determine a and b.

ANS--> We have taken the data and made it into a matrix form Ax=B this will enable us to get the most precise
a,b for the equation y = ax + b by the least square approximation method (learnt in MA106)
a = 0.324 (rounded off from the terminal output)
b = -1.094 (rounded off from the terminal output)

Assume now that you have determined the model described above. You see a person of height x. You can use the model to predict this person's height. Let your prediction be denoted by ˆy. When you measure the actual weight, now denoted by y, what can you say about the difference between the true height y and your intelligent guess ˆy using the linear model.

ANS--> The data can't never match but since we used least square approx. method, the errors will be close to zero
and hence for all practicle purpose we can take the values nearly the same but since it is just a guess
there are chances of this data to be wrong.However, we can also comment on the errors obtained , specifically, mean should be very less (say 0.05%) as positive and negative errors gets balanced . about the standard deviation , we can say thatit will be close to zero but will be more than mean (say less than 5%).
We can also see the actual means and standard deviations (in % of predicted values of y) of all the three cases of x.