HW1 Solution by Tianze Wang

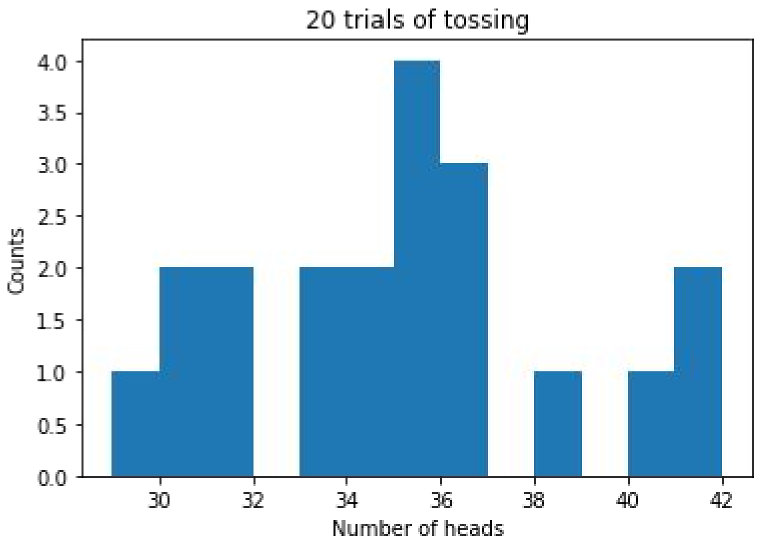
USCID:3993275888

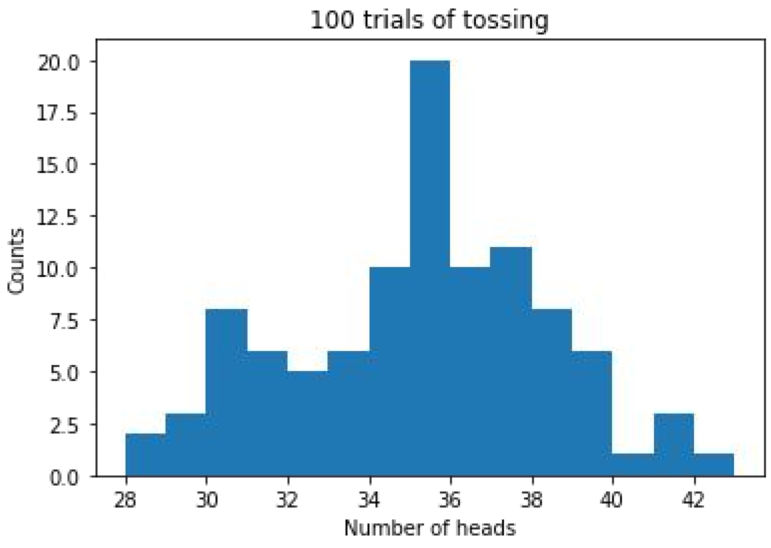
Q1(a):

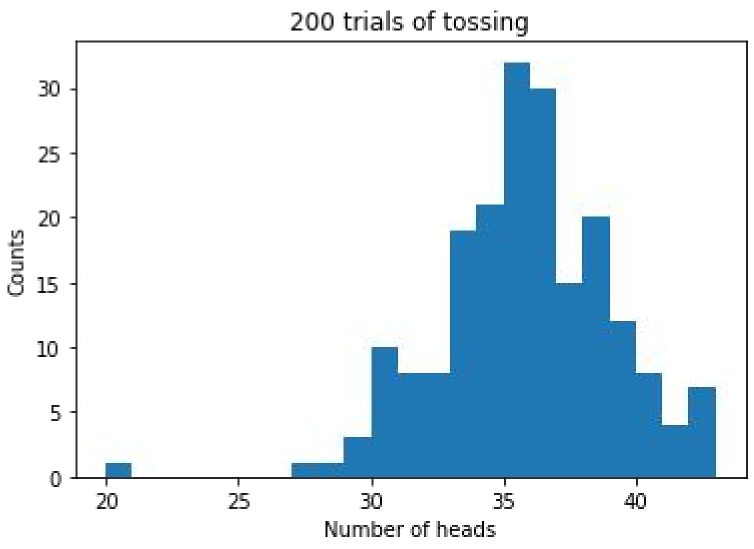
Number of heads: 34

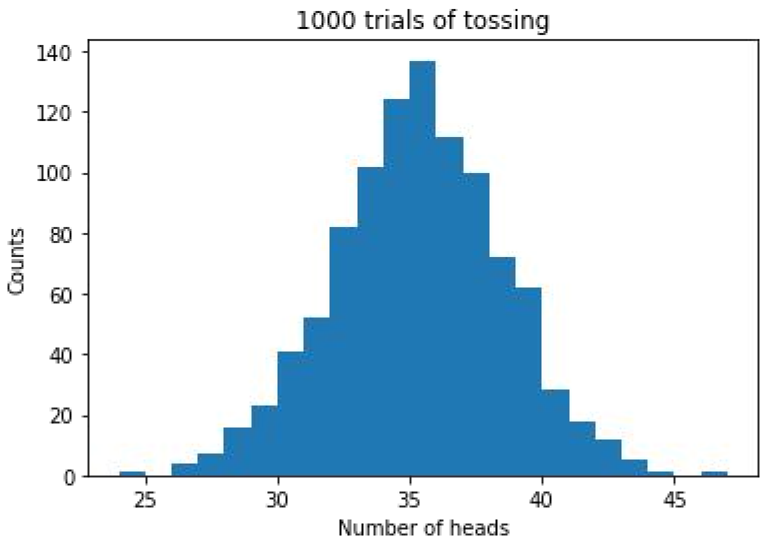
Longest run of heads: 7

Q1(b):



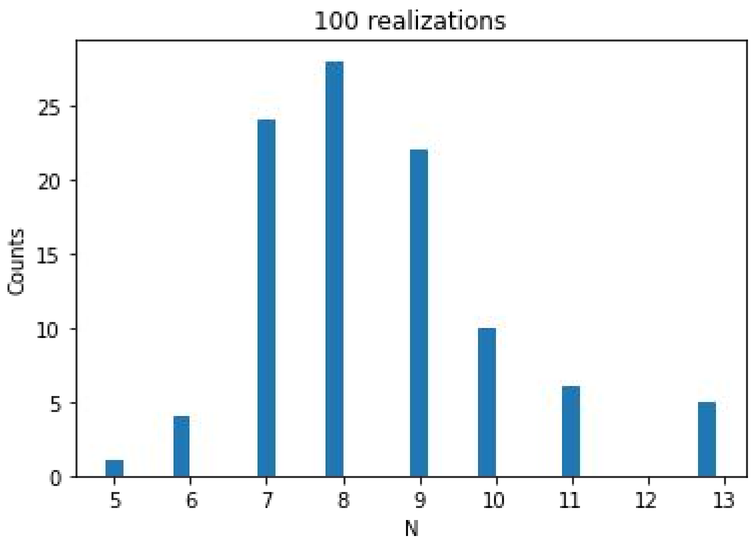


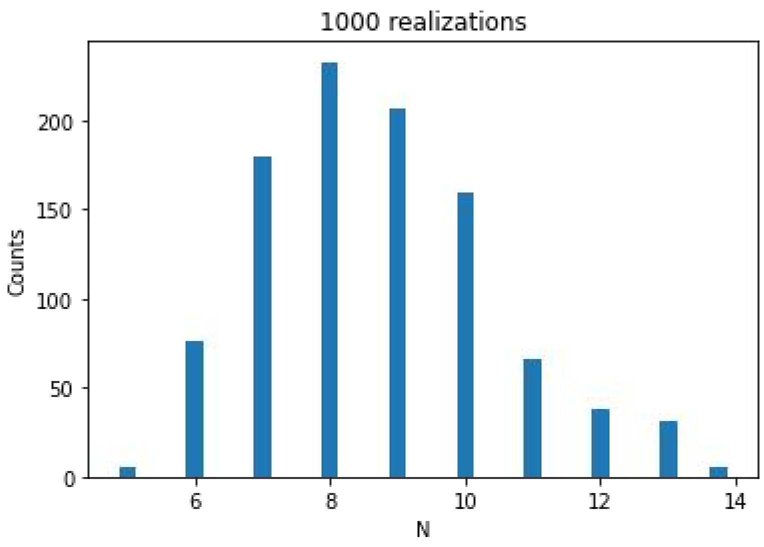


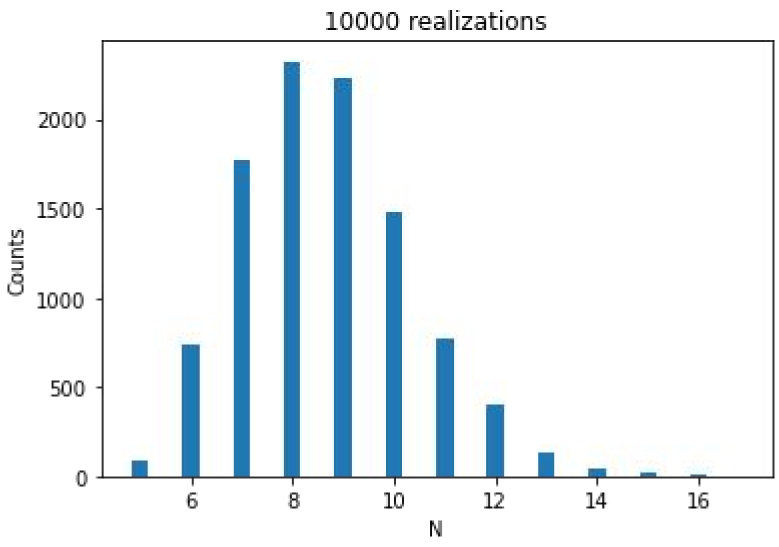


With the increament of experiment time, the histogram looks more like Gaussian Distribution with expectation of 35.

Q1(c):

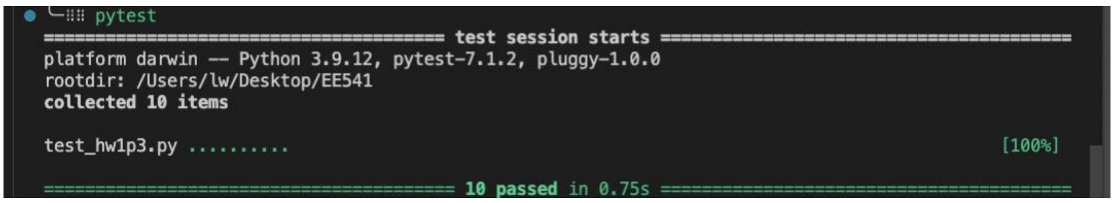






E[N] = 4 / (1/2*(1-0)) = 8

Q3:



```
● └─⠿ pytest
======================================= test session starts ========================================
platform darwin -- Python 3.9.12, pytest-7.1.2, pluggy-1.0.0
rootdir: /Users/lw/Desktop/EE541
collected 10 items

test_hw1p3.py ..........                                                                     [100%]

======================================== 10 passed in 0.75s ========================================
```

Code

Q1:

```
import matplotlib.pyplot as plt
import random

def toss():
    if random.random() < 0.7:
        return "Head"
    return "Tail"

count = 0
longest_run = 0
cur_longest = 0
for i in range(50):
    if toss() == "Head":
        count=count+1
        cur_longest = cur_longest + 1
    else:
        cur_longest = 0
    longest_run = max(longest_run, cur_longest)

print("Number of heads: " + str(count))
print("Longest run of heads: " + str(longest_run))

def toss2():
    if random.random() < 0.7:
        return 1
    return 0

def q1b(num):
    result = [0] * num
    for i in range(num):
        cur_case = []
        for n in range(50):
            cur_case.append(toss2())
        result[i] = sum(cur_case)
    return result

plt.figure()
plt.title("20 trials of tossing")
plt.xlabel("Number of heads")
plt.ylabel("Counts")
res = q1b(20)
edge_left = min(res)
edge_right = max(res)
num_bins = edge_right - edge_left
trial_20 = plt.hist(res, bins = num_bins)
plt.figure()
plt.title("100 trials of tossing")
plt.xlabel("Number of heads")
plt.ylabel("Counts")
res = q1b(100)
edge_left = min(res)
edge_right = max(res)
num_bins = edge_right - edge_left
trial_50 = plt.hist(res, bins = num_bins)
plt.figure()
plt.title("200 trials of tossing")
plt.xlabel("Number of heads")
plt.ylabel("Counts")
res = q1b(200)
edge_left = min(res)
edge_right = max(res)
num_bins = edge_right - edge_left
trial_200 = plt.hist(res, bins = num_bins)
plt.figure()
plt.title("1000 trials of tossing")
plt.xlabel("Number of heads")
plt.ylabel("Counts")
res = q1b(1000)
edge_left = min(res)
edge_right = max(res)
num_bins = edge_right - edge_left
trial_1000 = plt.hist(res, bins = num_bins)

def q1c(num):
    result = []
    for i in range(num):
        cur_longest = 0
        for n in range(50):
            if toss2() == 1:
                cur_longest = cur_longest + 1
            else:
                if cur_longest != 0:
                    result.append(cur_longest)
```

```
                        cur_longest = 0
        return result

plt.figure()
plt.title("500 trials of tossing")
plt.xlabel("Longest head run")
plt.ylabel("Counts")
res = q1c(500)
edge_left = min(res)
edge_right = max(res)
num_bins = edge_right - edge_left
trial_500 = plt.hist(res, bins = num_bins)
```

Q2:

```
import matplotlib.pyplot as plt
import random

def q2(num):
        result = [0]*num
        for i in range(num):
                N = 0
                cur_sum = 0
                while cur_sum <= 4:
                        cur_sum = cur_sum + random.random()
                        N = N + 1
                result[i] = N
        edge_left = min(res)
        edge_right = max(res)
        num_bins = edge_right - edge_left + 1
        plt.figure()
        plt.title(str(num) + " realizations")
        plt.xlabel("N")
        plt.ylabel("Counts")
        trial = plt.hist(result, bins = num_bins,align = 'left')

q2(100)
q2(1000)
q2(10000)
```

Q3:

```
from func import f
import sys

a = sys.argv[1]
```

```
b = sys.argv[2]

# Prerequisite check
try:
  float(a)
  float(b)
except:
  sys.stderr.write("Range error\n")
  # print('Range error', file=sys.stderr)
  sys.exit()
else:
  a = float(a)
  b = float(b)

if a >= b:
  sys.stderr.write("Range error\n")
  # print('Range error', file=sys.stderr)
  sys.exit()
if f(a)*f(b) >= 0:
  sys.stderr.write("Range error\n")
  # print('Range error', file=sys.stderr)
  sys.exit()

#Secant inplementation
CONV_CRIT = 1e-10

x0 = a
x1 = b
N = 0
while True:
  x2 = x1 - f(x1)*(x1-x0)/(f(x1) - f(x0))
  N += 1
  if(abs(x2 - x1) < CONV_CRIT):
    print(str(N), file=sys.stdout)
    print(str(x0), file=sys.stdout)
    print(str(x1), file=sys.stdout)
    print(str(x2), file=sys.stdout)
    sys.exit()
  x0 = x1
  x1 = x2
```