Lab2.1- Nhi Nguyen

COIN TOSSING

##1. a.

```
prob.heads = 0.6
num.tosses = 5
outcomes = sample(c(0,1), size = num.tosses, prob = c(1-prob.heads, prob.heads), replace = TRUE)
table(outcomes)
outcomes
total.heads = sum(outcomes)
total.heads
```

b. 0 is represented for tails, and 1 is for heads.

c. Sample() function with replacement (like replace = TRUE) means that the events are independent with the others and don't relate the other ones because after 1 event occurs like you withdraw a ball from a bag, then you put the another one in the gap here for the next occurence of event 2, so 2 different events have the same sample to randomly appear.

d. Using 0 and 1 to show the outcomes because the next step is required to calculate the times event happens like we can do the addition among $1 + 1 + 0 + 1 + \ldots$ So if we use T or H, we again have to do a "if-else" loop to find out whether it is T for plus 0 or it is H to plus 1.

e. I think it is reasonable to expect the differences in results because they happen randomly with replacement, but just in every single time because in conclusion we sum up all the results and have the same again.

##2. a.

```
prob.heads = 0.6
num.tosses = 5
num.replicate = 50
outcomes = vector("numeric", num.replicate)
set.seed(2018)
for (k in 1:num.replicate) {
  outcomes.replicate = sample(c(0,1), size = num.tosses, prob = c(1-prob.heads, prob.heads), replace = T
  print(outcomes.replicate)
  outcomes[k] = sum(outcomes.replicate)
}
outcomes
addmargins(table(outcomes))
```

b. there are 2 heads in the fourth replicate of the experience.

c.d.

```
heads.3 = (outcomes == 3)
table(heads.3)
```

20 out of 50 replicates we have exactly 3 heads .

e.

```
prob.heads = 0.6
num.tosses = 5
num.replicate = 100
outcomes = vector("numeric", num.replicate)
set.seed(2021)
for (k in 1:num.replicate) {
  outcomes.replicate = sample(c(0,1), size = num.tosses, prob = c(1-prob.heads, prob.heads), replace = 
  print(outcomes.replicate)
  outcomes[k] = sum(outcomes.replicate)
}
outcomes
addmargins(table(outcomes))
heads.3 = (outcomes == 3)
table(heads.3)
```

0.3514

f. 0.3456

g. the results we've got from ques d and e is nearly the same as one in ques f as it is based on the sample we choose for the event. the bigger the sample is, the more reliable the value is. so when calculating from 1000 replicates, it may be more trusted than the one in d and less than the one in e (50<1000<10000)

Mandatory Drug Testing

##3. 1 - (1-0.012)^150 = 83.64%

##4 a. i. they will be tested positive (but falsely) or negative ii. iii. we use 1 to describe the occurrence of falsely positive people and 0 for the negative, as we can make an addition among the occurrences by sum of a lot of 1, so we can calculate more easily

b.c.

```
prob.positive = 0.012
num.test = 150
num.replicate = 100000
positive = vector("numeric", num.replicate)
set.seed(2021)
for (k in 1:num.replicate) {
  pos.replicate = sample(c(0,1), size = num.test, prob = c(1-prob.positive, prob.positive), replace = T
  positive[k] = sum(pos.replicate)
}
positive
table(positive)
```

```
addmargins(table(positive))
pos.more = (positive >= 1)
table(pos.more)
```

there was slightly more than 1 value represented for employee who tested positively

###Mammograms ##5 a. the specificity describes the possibility of that test to go on the correct way (negative test) whereas the prob of the previous ques shows the risk of that event when going wrong (falsely positive), so two variables are understood in 2 different ways. however, both of them are used to calculate the probability of any event.

b.

```
prob.neg = 0.95
prob.falsepos = 0.05

num.mammo = 50
num.rep = 1000

falsepos = vector("numeric", num.rep)

for (k in 1:num.rep) {
  outcomes = sample(c(0,1), size = num.mammo, prob = c(prob.neg, prob.falsepos), replace = TRUE)
  falsepos[k] = sum(outcomes)
}
res = (falsepos<=1)
addmargins(table(res))
```

c.

```
prob.neg = 0.99
prob.falsepos = 0.01

num.mammo = 50
num.rep = 1000

falsepos = vector("numeric", num.rep)

for (k in 1:num.rep) {
  outcomes = sample(c(0,1), size = num.mammo, prob = c(prob.neg, prob.falsepos), replace = TRUE)
  falsepos[k] = sum(outcomes)
}
res = (falsepos<=1)
addmargins(table(res))
```