CS 237 Fall 2019 Homework Three

Due date: PDF file due Thursday September 26 @ 11:59PM in GradeScope with 6-hour grace period

Late deadline: If submitted up to 24 hours late, you will receive a 10% penalty (with same 6 hours grace period)

General Instructions

Please complete this notebook by filling in solutions where indicated. Be sure to "Run All" from the Cell menu before submitting.

There are two sections to the homework: problems 1 - 8 are analytical problems about last week's material, and the remaining problems are coding problems which will be discuss in lab next week.

In [1]:

```
# Here are some imports which will be used in code that we write for CS 237

# Imports used for the code in CS 237

import numpy as np                  # arrays and functions which operate on array
import matplotlib.pyplot as plt     # normal plotting
import seaborn as sns               # Fancy plotting
import pandas as pd                 # Data input and manipulation

from math import log,pi                  # or just import math and use anything th
ere, conflicts with numpy?
from numpy.random import random, randint, uniform, choice, shuffle, seed
from collections import Counter

%matplotlib inline

# Use this next function to round to 4 digits.
# If the number is less than 0.00005 this will
# just produce 0.0000 so just print it normally.

def round4(x):
    return round(x+0.00000000001,4)

def round4_list(L):
    return [ round4(x) for x in L ]


# Useful code from HW 01

# This draws a useful bar chart for the distribution of the
# list of integers in outcomes

def show_distribution(outcomes, title='Probability Distribution', my_xticks =
[], f_size = (8,6)):
    plt.figure(figsize=f_size)
    num_trials = len(outcomes)
    X = range( int(min(outcomes)), int(max(outcomes))+1 )
    freqs = Counter(outcomes)
    Y = [freqs[i]/num_trials for i in X]
    plt.bar(X,Y,width=1.0,edgecolor='black')
    if my_xticks != []:
        plt.xticks(X, my_xticks)
    elif (X[-1] - X[0] < 30):
        ticks = range(X[0],X[-1]+1)
        plt.xticks(ticks, ticks)
    plt.xlabel("Outcomes")
    plt.ylabel("Probability")
    plt.title(title)
    plt.show()

# This function takes a list of outcomes and a list of probabilities and
# draws a chart of the probability distribution.
# It allows labels for x axis with numbers or strings; for the latter, you
# still need to give the numeric labels, but can overwrite them with your string
labels.

def draw_distribution(Rx, fx, title='Probability Distribution', my_xticks = [],
f_size = (8,6)):
    plt.figure(figsize=f_size)
    plt.bar(Rx,fx,width=1.0,edgecolor='black')
    plt.ylabel("Probability")
    plt.xlabel("Outcomes")
    if my_xticks != []:
        plt.xticks(Rx, my_xticks)
    elif (Rx[-1] - Rx[0] < 30):
        ticks = range(Rx[0],Rx[-1]+1)
        plt.xticks(ticks, ticks)
    plt.title(title)
    plt.show()
```

Analytical Problems Introduction

Some of these problems concern the familiar experiment of throwing two dice and counting the number of dots that show on both. It is always a good idea to draw the search space if possible, so here is a diagram such as I put on the board; this also shows you how to use HTML inside Markdown cells!

Outcomes from Two Dice

	1	**2**	**3**	**4**	**5**	**6**
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Problem 1 (Conditional Probability)

Suppose you roll two fair dice and count the number of dots showing. What is the probability of a sum of 5 if

(a) the second roll is not 3?

(b) they land on different numbers?

Solution:

```
(a)P(sum of two fair dice is 5 | the second roll is not 3) = 3/30 = 0.1000

(b)P(sum of two fair dice is 5 | they land on different numbers) = 4/30 =
 0.1333
```

Problem 2 (Conditional Probability)

Suppose we have two events A and B. For each of these three cases, give the numeric value or the simplest formula for $P(A \mid B)$:

(a) $A \cap B = \emptyset$ (A and B are disjoint)

(b) $A \subset B$

(c) $B \subset A$

Solution:

```
(a)P(A|B) = P(A n B) / P(B). If P(A n B)==0, P(A|B) = 0.

(b)When A is a subset of B, it means that all values of A are also in B. T
herefore, P(A n B) == P(A) since all values existing in set A are all in s
et B. Therefore, P(A n B) = P(A)/P(B).

(c)When B is a subset of B, it means that all values of B are also in A. T
herefore, P(A n B) == P(B) since all values existing in set B are also in
 set A. Therefore, P(A n B) = P(B)/P(B) = 1.
```

Problem 3

(Based on a true story, more or less....)

Wayne falls asleep watching an old Clint Eastwood movie at 4am after preparing his lecture for CS 237...

He is lying on the sidewalk after robbing a bank, in pain and mulling over how to quantify the uncertainty of his survival, when Dirty Harry walks over. Dirty Harry pulls out his 44 Magnum and puts two bullets opposite each other in the six slots in the cylinder (e.g., if you number them 1 .. 6 clockwise, he puts them in 1 and 4), spins the cylinder randomly, and, saying "The question is, are you feeling lucky, probabalistically speaking, computer science punk?" points it at Wayne's head and pulls the trigger.... "CLICK!" goes the gun (no bullet) and Dirty Harry smiles... "How about that .... Let's see if this gun is memory-less!" Without spinning the cylinder again, he points the gun at Wayne's head and pulls the trigger again.

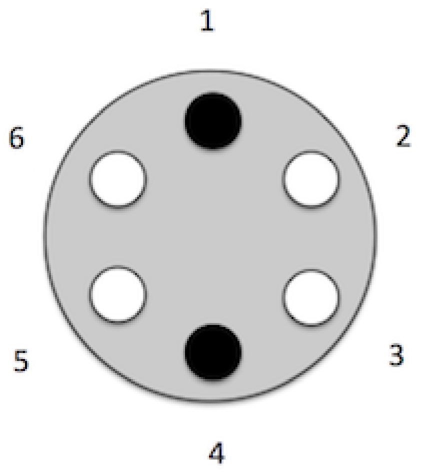
(a) What is the probability that (at least in my dream) you will have to have another instructor finish out CS 237 (because Wayne has a really BAD headache and can't continue)?

(b) Now, suppose that when Dirty Harry put the bullets in the gun, he put them right next to each other (e.g., in slots 1 and 2). He spins it as usual. What is the probability in this case that you will have another instructor finish teaching CS 237?

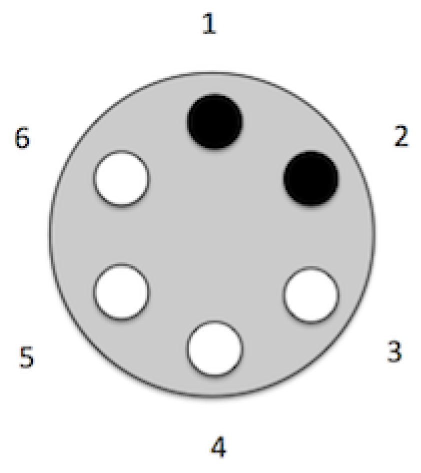
(c) Suppose Dirty Harry puts the bullets in two random positions in the cylinder and we don't have any idea where they are. He spins it as usual. Now what is the probability that I will not be able to finish teaching CS 237?

Hint: This has nothing to do with the memory-less property (Dirty Harry never took CS 237) .... and we could solve it by just considering what the probability is for each of the patterns (no bullet, no bullet) and (no bullet, bullet) as we go around the circle of slots in the cylinder.

Here is what the cylinder might look like in case (a) before Dirty Harry randomizes the positions by spinning it:



Here is what it might look like in case (b) before being randomized:



Solution:

```
(a) P(another instructor finish out CS 237) = 2/4 = 0.5000.

(b) P(another instructor finish out CS 237) = 1/4 = 0.2500.

(c) P(not be able to finish teaching CS 237) = 2/5 = 0.4000.
```

Problem 4 (Independence)

Suppose you roll two dice and count the number of dots showing. Let A = "the sum of the dots showing on the two rolls is a prime number."

(a) Let B = "The first toss showed an even number of dots." Are A and B independent? Show your work.

(b) Let C = "the second toss was greater than the first." Are A and C independent? Show your work.

(c) Suppose we pick a value k for $1 \leq k \leq 6$ and let event D$_k$ = "the first toss was greater than or equal to k." For which values of k are A and D$_k$ independent? Show your work.

Solution:

```
(a) P(A) = P(sum of dots on two rolls is a prime number) = P(sum is 2,3,5,
7,or 11) = P(sum is 2) + P(sum is 3) + P(sum is 5) + P(sum is 7) + P(sum i
s 11) = 1/36 + 2/36 + 4/36 + 6/36 + 2/36 = 15/36 = 5/12.
    P(B) = P(First toss showed an even number) = P(first toss is 2,4,or 6)
= 1/2 * 1 = 1/2.
    P(A n B) = P(First toss is even, then second toss has to sum up to pri
me number) = P(when first toss is 2) + P(when first toss is 4) + P(when fi
rst toss is 6) = 1/6 * 3/6 + 1/6 * 2/6 + 1/6 * 2/6 = 3/36 + 2/36 + 2/36 =
 7/36
    P(A)* P(B) = 5/12 * 1/2 = 5/24.
    Since P(A)* P(B) does not equal P(A n B), events A and B are not indep
endent.

(b) P(A) = 5/12.
    P(C) = P(second toss was greater than the first) = P(first toss is 1)
 + P(first toss is 2) + P(first toss is 3) + P(first toss is 4) + P(first
 toss is 5) + P(first toss is 6) = 1/6 * 5/6 + 1/6 * 4/6 + 1/6 * 3/6 + 1/6
* 2/6 + 1/6 * 1/6 + 1/6 * 0 = 5/36 + 4/36 + 3/36 + 2/36 + 1/36 + 0 = 15/36
= 5/12.
    P(A n C) = P(second toss was greater than the first and sum of dots on
two rolls is a prime number) = P(first toss is 1) + P(first toss is 2) + P
(first toss is 3) + P(first toss is 4) + P(first toss is 5) + P(first toss
is 6) = 1/6 * 3/6 + 1/6 * 2/6 + 1/6 * 1/6 + 1/6 * 0 + 1/6 * 1/6 + 1/6 * 0
```

Problem 5 (Independence)

For three events A, B, and C, we know that

- A and C are independent,
- B and C are independent,
- A and B are disjoint,

Furthermore, suppose that $P(A \cup C) = \frac{2}{3}$, $P(B \cup C) = \frac{3}{4}$, $P(A \cup B \cup C) = \frac{11}{12}$.

Find $P(A)$, $P(B)$, and $P(C)$.

Solution:

```
P(A) * P(C) = P(A n C).
P(B) * P(C) = P(B n C).
P(A n B) = empty Set.
P(A U C) = P(A) + P(C) - P(A n C) = 2/3.
P(A n C) = P(A) + P(C) - 2/3.
P(B U C) = P(B) + P(C) - P(B n C) = 3/4.
P(B n C) = P(B) + P(C) - 3/4.
Since A and B are disjoint, P(A n B) = 0.
Therefore, events A, B, and C are also disjoint, giving P(A n B n C) = 0.
P(A U B U C) = P(A) + P(B) + P(C) - P(A n C) - P(A n B) - P(B n C) - P(A n
B n C) = 11/12.
11/12 = P(A) + P(B) + P(C) - P(A) - P(C) + 2/3 - 0 - P(B) - P(C) + 3/4 -
 0.
11/12 = -P(C) + 2/3 + 3/4 = -P(C) + 17/12.
-6/12 = -P(C).
P(C) = 1/2 = 0.5000.

Given that P(C) is 0.5,
P(A U C) = P(A) + P(C) - P(A n C) = P(A) + P(C) - P(A) * P(C) = P(A) + 0.5
- 0.5 * P(A) = 2/3.
P(A) - 0.5 * P(A) = 2/3 - 1/2 = 1/6.
0.5 * P(A) = 1/6.
P(A) = 2/6 = 0.3333.

Given that P(C) is 0.5,
P(B U C) = P(B) + P(C) - P(B n C) = P(B) + P(C) - P(B) * P(C) = P(B) + 0.5
- 0.5 * P(B) = 3/4.
P(B) - 0.5 * P(B) = 3/4 - 1/2 = 1/4.
0.5 * P(B) = 1/4.
P(B) = 2/4 = 0.5000.
```

Problem 6

You randomly shuffle a 52-card deck of cards. We will consider what happens as we draw 3 cards from the deck to form a sequence of cards (for a - c) or a set of cards (d).

Answer each of the following questions, showing all relevant calculations. You should analyze these using tree diagrams, but no need to show the diagrams in your answer.

(a) Suppose you draw three cards from the deck with replacement; what is the probability that the first card is red, the second is a spade, and the third is a facecard (Jack, Queen, or King)? [Hint: these are ordered!]

(b) Suppose you draw three cards from the deck with replacement; what is the probability that the first and third cards have the same color, but the second is a different color?

(c) Repeat (b), but without replacement (cards are not put back). [Hint: a tree diagram might be useful.]

Solution:

(a) P(first red, second spade, third facecard) = 26/52 ***13/52*** 12/52 = 1/2 ***1/4*** 3/13 = 0.02885

(b) P(first and third cards have same color, second color different) = P(first and third Black and second Red) + P(first and third Red and second Black) = 1/2 ***1/2*** 1/2 + 1/2 ***1/2*** 1/2 = 1/8 + 1/8 = 1/4 = 0.25

(c) P(first and third cards have same color, second color different) = P(first and third Black and second Red) + P(first and third Red and second Black) = 26/52 ***26/51*** 25/50 + 26/52 ***26/51*** 25/50 = 0.2549

Problem 7

You have two bags of colored balls. Bag A contains 3 red and 2 black balls; and Bag B contains 1 red ball and 5 black balls. Both bags are shaken so that they are in random order but the bags are kept separate.

Answer each of the following questions, showing all relevant calculations. If necessary, analyze these using a tree diagram, but no need to show the diagram in your answer.

(a) Suppose you draw a ball from Bag A and find it is red; then you draw a ball from Bag B and find that it is black; you throw all the balls from both bags into a third bag, shake it, and draw a ball. What is the probability that it is black?

(b) Suppose you flip a fair coin to choose one of the bags, and then draw one ball. What is the probability that it is red?

(c) Suppose you tell a friend to flip a fair coin to choose one of the bags, and then draw a ball without telling you which bag it came from; you see that it is red. What is the probability that it came from Bag A?

Solution:

```
(a) P(black ball comes from third bag) = 4/9 = 0.4444
(b) P(red ball) = P(flop comes out head and choose from A) + P(flop comes
 out tail and choose from B) = 1/2 * 3/5 + 1/2 * 1/6 = 0.3 + 0.0833 = 0.38
33
(c) P(came from Bag A | red ball) = (1/2 * 3/5) / (4/11) = 0.825
```

Problem 8 (Circular Permutations)

In how many ways can 7 people { A, B, C, D, E, F, G } be seated at a round table if

(a) A and B must not sit next to each other;

(b) C, D, and E must sit together (i.e., no other person can sit between any of these three)?

(c) A and B must sit together, but neither can be seated next to C or D.

Consider each of these separately. For (c) you may NOT simply list all possibilities, but must use the basic principles we have developed (you may check your work with a list if you wish).

Hint: Conceptually, think of the groups of two or three people as one "multi-person" entity in the overall circular arrangement. However, a "multiperson" is an unordered entity, and you will have to think about how many ways a "multiperson" could be ordered. It may help to draw a diagram, fixing a particular person at the top of the circle (thereby eliminating the duplicates due to rotations).

Solution:

```
   (a) Since it is circular permutation, there are total of (7-1)! = 6! =
 720 ways to be seated at a round table. If A and B must not sit next to e
ach other, we need to subtract the ways they can sit together from the 720
ways we calculated. Since sitting next to each other can count as a pair
 (AB), it is same as 6 people sitting at a round table. If we calculate ho
w many ways can 6 people be seated, it is (6-1)! = 5! = 120 ways. However,
we need to consider AB and BA as a different way of sitting, multiplying 1
20 by 2, giving 240. Subtracting 240 from 720, we get 480. There are 480 w
ays so that A and B must not sit next to each other.

   (b) Since C,D, and E must sit together, we can group CDE as one unit, g
iving 5 people to be seated at a round table. Since it is circular permuta
tion, there are (5-1)! = 4! = 24 ways they can be seated together. Howeve
r, CDE can be seated in whatever order they want. They can be seated CDE o
r DEC or etc. There are total 3! ways this is possible. 24 * 3! = 24 * 6 =
144 ways. There are 144 ways so that C, D, and E must sit next to each oth
er.

   (c) Since it is circular permutation where A and B should be next to ea
ch other, if you count A and B as a same unit, AB or BA, there are total o
f (6-1)! * 2 = 5! * 2 = 120 * 2 = 240 ways to be seated at a round table.
 Things that are not allowed is AB sitting next to C or D. Consider there
 are 6 units (one is AB) that should not sit next to C or D. We can calcul
ate it by calculating the ways AB can be next to C or D and subtracting it
from 240 ways that we calculated. We can consider AB C as one unit and the
refore, there are (5-1)! = 4! = 24 ways. Since the order can be C AB or AB
C in a round table, there are 2*24 = 48 ways. Same works with D. There are
48 ways AB is next to D. Subtracting both of them from 240, it is 240 - 48
-48 = 144 ways. There are 144 ways so that A and B are next to each other
 while not next to C or D.
```

Lab Introduction: Poker Probability

In this lab we will explore Poker Probability, which is calculating the probability of various hands in the game of poker. This is, again, exploring how to confirm our theoretical understanding with experiments. If our experiments, as we increase the number of trials, converge to our theoretical calculation, then we have almost certainly analyzed it correctly.

There are many versions of poker (see here (http://www.wikihow.com/Play-Poker)) but the game we will study is called "five-card draw." It is described (https://www.pokernews.com/strategy/5-card-draw-rules-how-to-play-five-card-draw-poker-23741.htm) as follows:

> Once everyone has paid the ante, each player receives five cards face down. A round of betting then occurs. If more than one player remains after that first round of betting, there follows a first round of drawing. Each active player specifies how many cards he or she wishes to discard and replace with new cards from the deck. If you are happy with your holding and do not want to draw any cards, you “stand pat.” Once the drawing round is completed, there is another round of betting. After that if there is more than one player remaining, a showdown occurs in which the player with the best five-card poker hand wins.

Here is an excellent short YT video on the basics of Poker: YT (https://www.youtube.com/watch?v=xfqMC3G37VE)

The only part we will care about is the final calculation of which hand wins: basically, the least probable hand wins. When you learn poker, then, one of the first things you have to learn is the ordering of the hands from most to least likely. ***Poker probability*** refers to calculating the exact probabilities of hands. The Wikipedia article (https://en.wikipedia.org/wiki/Poker_probability) contains the exact results and the formulae used to calculate them.

In this lab we will develop a framework for dealing 5-card hands and empirically estimating the probabilites of various hands. In fact, we will be able to do nearly all the hands commonly encountered. Our only constraint is that for the rarest hand, a Royal Flush, since there are only 4 such hands, the probability is so small it would take too long to get a reasonable estimate, and so we shall ignore this case.

This lab should help your understanding of the counting techniques covered in lecture.

Preface: Card Games and Probability

First we will first explore how to encode a standard deck of 52 playing cards, how to perform various tests on cards, and how to deal hands. To remind you, here is the illustration showing all the cards: cards (http://www.cs.bu.edu/fac/snyder/cs237/images/PlayingCards.png).

In [2]:

```
# We will represent cards as a string, e.g., 'AC' will be Ace of Clubs

# Denominations: 2, ..., 10,  'J' = Jack, 'Q' = Queen, 'K' = King, 'A' = Ace
Denominations = ['2','3','4','5','6','7','8','9','10','J','Q','K','A']

# Suits 'S' = Spades, 'H' = Hearts, 'D' = Diamonds, 'C' = Clubs
Suits = ['C', 'H', 'S', 'D']

# Note that colors are determined by the suits (hearts and diamonds are red, oth
ers black,
# so, AC is Black

# List comprehensions are a great way to avoid explicit for loops when creating
 lists

Deck =  [(d+s) for d in Denominations for s in Suits]   # Note the double for lo
op

print( Deck )
```

```
['2C', '2H', '2S', '2D', '3C', '3H', '3S', '3D', '4C', '4H', '4S',
'4D', '5C', '5H', '5S', '5D', '6C', '6H', '6S', '6D', '7C', '7H', '7
S', '7D', '8C', '8H', '8S', '8D', '9C', '9H', '9S', '9D', '10C', '10
H', '10S', '10D', 'JC', 'JH', 'JS', 'JD', 'QC', 'QH', 'QS', 'QD', 'K
C', 'KH', 'KS', 'KD', 'AC', 'AH', 'AS', 'AD']
```

In [3]:

```
# Now we can "deal" cards by choosing randomly from the deck

seed(0)                          # seed makes sure that all your computations s
tart with the same random sequence;
                                 # this not really important, and only necessar
y for debugging and grading.
def dealCard():
    return choice(Deck)          # choice randomly chooses an element of a list

print( dealCard() )
```

```
KC
```

In [4]:

```
# When dealing a hand in cards, the selection of cards is without replacement, t
hat is, cards are removed from
# the deck one by one and not put back. This can be simulated in the choice func
tion by setting the replace
# parameter to False.

seed(0)

def dealHand(withReplacement = False,size = 5):
    return choice(Deck,size,withReplacement)                     # chooses a list
 of size elements

print( dealHand() )
```

```
['9C' 'JH' '4D' '10S' '2S']
```

In [5]:

```
# extract the denomination and the suit from a card

def denom(c):
    return c[0:-1]

def suit(c):
    return c[-1]

# The function rank(c) will simply return the position of the card c PLUS 2 in t
he list 2, 3, ...., K, A. This will be used in an essential
# way in our code below. Although in the diagram given lecture, Ace is below 2,
 the Ace is actually considered to be ordered
# above the King, for example in determining a straight, under "Ace high rules."

#  rank(2) = 2, ...., rank(10) = 10, rank(Jack) = 11, rank(Queen) = 12, rank(Kin
g) = 13, rank(Ace) = 14

def rank(c):
    return Denominations.index(denom(c))+2

# Now we want to identify various kinds of cards

def isHeart(c):
    return ( suit(c) == 'H')

def isDiamond(c):
    return ( suit(c) == 'D')

def isClub(c):
    return ( suit(c) == 'C')

def isSpade(c):
    return ( suit(c) == 'S')

def isRed(c):
    return ( isHeart(c) or isDiamond(c) )

def isBlack(c):
    return (not isRed(c))

def isFaceCard(c):
    return rank(c) >= 11 and rank(c) <= 13
```

Example Problem: What is probability that a 5-card hand has exactly 3 red cards?

Remember that in finite probability, for any event A,

$$P(A) = \frac{|A|}{|S|}.$$

Therefore, what we need to do in problems involving the probability of various kinds of hands in card games is to count the number of possible such hands, and divide by the total number of all possible hands. We developed analytical tools in lecture to do this, but here we are going to estimate it with repeated trials of dealing hands and testing for a given kind of hand.

In general for all but the last problem, we will use 100,000 trials to get a reasonable estimate of the probability. Since 1/100000 = 0.00001 this means our resolution for experimental probabilities is 5 decimal places.

In [6]:

```
# Return True iff the number of red cards in the hand h is 3
def threeRed(h):
    redCards = [c for c in h if isRed(c)]
    return (len(redCards) == 3)

seed(0)

# Run the experiment for 10,000 trials
# Print out probability that a 5-card hand has exactly 3 red cards

num_trials = 10**5
trials = [dealHand() for k in range(num_trials)]       # create list of 10000 ha
nds randomly dealt

if(num_trials <= 10):       # Just for this example, you don't need to do this u
nless you are debugging
    print(trials)

hands =  [threeRed(h) for h in trials]                 # convert this to list of
true and false values

if(num_trials <= 10):
    print(hands)

prob = hands.count(True) / num_trials                  # count the number of Tru
e values and divide by num_trials

# probability for 100,000 trials should be close to analytical value of 0.3251

print('\nProbability of exactly 3 red cards in a 5-card hand is ' + str(prob))
```

```
Probability of exactly 3 red cards in a 5-card hand is 0.32225
```

In [7]:

```
# here is another way to do it, but it is a bit cryptic!

# trials was calculated above

seed(0)

prob2 = sum( [1 for h in trials if threeRed(h)] ) / num_trials

print('\nProbability of exactly 3 red cards in a 5-card hand is ' + str(prob2))
```

```
Probability of exactly 3 red cards in a 5-card hand is 0.32225
```

In [8]:

```
# If you like cryptic, then you can put it all, including the calculation of tri
als in one line!
# Here are two examples. Note: these run the experiment again,  so they won't ma
tch the previous 2 answers precisely.

seed(0)

prob3 = sum( [1 for h in [dealHand() for k in range(num_trials)] if threeRed(h)]
) / num_trials

print('\nProbability of exactly 3 red cards in a 5-card hand is ' + str(prob3))

seed(0)

prob4 = sum( [1 for k in range(num_trials) if threeRed(dealHand())] ) / num_tria
ls

print('\nProbability of exactly 3 red cards in a 5-card hand is ' + str(prob4))
```

```
Probability of exactly 3 red cards in a 5-card hand is 0.32225

Probability of exactly 3 red cards in a 5-card hand is 0.32225
```

Example: What is probability that a 5-card hand has at least 3 Diamonds?

In [9]:

```
# Print out probability that a 5-card hand has 3, 4, or 5 diamonds.

seed(0)

def atLeast3Diamonds(h):
    return (len([c for c in h if isDiamond(c)]) >= 3)

num_trials = 10**5


hands =  [atLeast3Diamonds(h) for h in trials]                # convert this to
list of true and false values
prob = hands.count(True) / num_trials

# Should be close to analytical value of 0.0928

print('Probability of at least 3 diamonds in a 5-card hand is ' + str( prob ))
```

```
Probability of at least 3 diamonds in a 5-card hand is 0.09278
```

Problem 9: What is probability of a flush in Poker?

In Poker, a ***flush*** is 5 cards of the same suit, but excludes straight flushes and royal flushes; these, however, are so rare (there are only 40 of them in all), that they are around or below our resolution (0.00001), so we just will determine if all suits are the same.

In [10]:

```
# Print out probability that a 5-card hand has all the same suit

seed(0)

num_trials = 10**5
def flush(h):
    if((len([c for c in h if isDiamond(c)]) == 5)):
        return True
    if((len([c for c in h if isClub(c)]) == 5)):
        return True
    if((len([c for c in h if isSpade(c)]) == 5)):
        return True
    if((len([c for c in h if isHeart(c)]) == 5)):
        return True
    else:
        return False

hands = [flush(h) for h in trials]
prob = hands.count(True) / num_trials
print('Probability of flush is ' + str(prob))
```

```
Probability of flush is 0.002
```

Problem 10: What is probability of a straight in Poker?

In poker, a ***straight*** a hand in which the ranks form a contiguous sequence, e.g., 2,3,4,5,6. The suits do not matter. Also, for simplicity, we will use the "deuce-to-seven low rules (https://en.wikipedia.org/wiki/List_of_poker_hands#Straight)" whereby the Ace must count as a high card (above the King). This simplifies the calculation a little bit, since we can just sort the cards by rank and check if they are contiguous.

In [11]:

```
# Print out probability that a 5-card hand is a straight

seed(0)

num_trials = 10**5

def straight(h):
    lst = []
    for x in h:
        y = denom(x)
        if(y == 'J'):
            lst.append(11)
        if(y == 'Q'):
            lst.append(12)
        if(y == 'K'):
            lst.append(13)
        if(y == 'A'):
            lst.append(14)
        if(y == '2'):
            lst.append(2)
        if(y == '3'):
            lst.append(3)
        if(y == '4'):
            lst.append(4)
        if(y == '5'):
            lst.append(5)
        if(y == '6'):
            lst.append(6)
        if(y == '7'):
            lst.append(7)
        if(y == '8'):
            lst.append(8)
        if(y == '9'):
            lst.append(9)
        if(y == '10'):
            lst.append(10)
    lst.sort()
    for x in range(len(lst) - 1):
        if (lst[x] + 1 != lst[x+1]):
            return False
    return True


hands = [straight(h) for h in trials]
probs = hands.count(True) / num_trials
print('Probability of straight is ' + str(probs))
```

```
Probability of straight is 0.0036
```

Problem 11: Rank Signature of a poker hand

Let us define the ***rank signature*** of a hand as an ordered histogram of the ranks occurring in the hand; that is, we count the frequency of the ranks occurring in the hand, and ***order*** this sequence. Here are some examples:

- Five cards all of different ranks (e.g., Ace, 4, 2, King, 8): `[1,1,1,1,1]`
- One pair, 2 cards of the same rank, and 3 more all of different ranks (e.g., 2,2,6,3,Ace): `[1,1,1,2]`
- Two pair, 2 pairs (of different ranks) and one card of a different rank (e.g., 2,2,Ace,3,Ace): `[1,2,2]`
- Full house, 2 cards of the same rank, and 3 cards of the same rank (e.g., 8,Jack,8,8,Jack): `[2,3]`

Many poker hand can be defined solely in terms of the ranks involved. The importance of this concept is that once we write a function to estimate the probability of a given signature, we can then immediately calculate the probability of many different poker hands.

For this problem you must write a function which calculate the probability that a 5-card hand has a given signature and verify it by calculating the probability of no two ranks being the same (first choose 5 different ranks, then consider all possible enumerations of suits):

$$\frac{\binom{13}{5} * 4^5}{\binom{52}{5}} = 0.5071$$

Note: This is NOT the same as "no pair/high card" since in Poker, this hand means you don't have a flush or a straight (both of which have 5 cards of different rank). To find out the correct probability, you would need to subtract the possibility of a flush or a straight (or both, a straight-flush), which we will do in the next problem.

In [12]:

```
#### Run the experiment for 100,000 trials
# Print out probability that a 5-card hand has a given signature

seed(0)

num_trials = 10**5

# returns the rank signature as a list
def has_rank_signature(h,rank_signature):
    lst = []
    lstTwo = []
    for x in h:
        y = denom(x)
        if(y == 'J'):
            lst.append(11)
        if(y == 'Q'):
            lst.append(12)
        if(y == 'K'):
            lst.append(13)
        if(y == 'A'):
            lst.append(14)
        if(y == '2'):
            lst.append(2)
        if(y == '3'):
            lst.append(3)
        if(y == '4'):
            lst.append(4)
        if(y == '5'):
            lst.append(5)
        if(y == '6'):
            lst.append(6)
        if(y == '7'):
            lst.append(7)
        if(y == '8'):
            lst.append(8)
        if(y == '9'):
            lst.append(9)
        if(y == '10'):
            lst.append(10)
    for x in range(15):
        if x in lst:
            count = 0
            for y in range(5):
                if (x == lst[y]):
                    count = count + 1
            if count != 0:
                lstTwo.append(count)
    lstTwo.sort()
    rank_signature.sort()
    if len(lstTwo) != len(rank_signature):
        return False
    else:
        for x in range(len(lstTwo)):
            if(lstTwo[x] != rank_signature[x]):
                return False
        return True

def probability_of_rank_signature(rank_signature,num_trials):
    hands = [has_rank_signature(h, rank_signature) for h in trials]
    prob = hands.count(True) / num_trials
    return prob
```

Problem 12: Using rank signature to calculate six different poker hands

Problem 12 (A): What is probability of No Pair/High Card in Poker?

For this, you must use the rank signature of 5 different ranks (as in the previous problem) but exclude any hands that are straights or flushes.

In [13]:

```
# probability should be close to analytical value of 0.5012

seed(0)
num_trials = 10**5
value = probability_of_rank_signature([1,1,1,1,1],num_trials) - prob - probs
print("Probability of No Pair/High Card in Poker is", value)
```

```
Probability of No Pair/High Card in Poker is 0.5031599999999999
```

Problem 12 (B): What is probability of One Pair in Poker?

In [14]:

```
# probability should be close to analytical value of 0.4226

seed(0)
num_trials = 10**5
value = probability_of_rank_signature([1,1,1,2],num_trials)
print("Probability of one pair Card in Poker is", value)
```

```
Probability of one pair Card in Poker is 0.42096
```

Problem 12 (C): What is probability of Two Pairs in Poker?

In [15]:

```
# probability should be close to analytical value of 0.047539

seed(0)
num_trials = 10**5
value = probability_of_rank_signature([1,2,2],num_trials)
print("Probability of two pairs in Poker is", value)
```

```
Probability of two pairs in Poker is 0.04783
```

Problem 12 (D): What is probability of Three of a Kind in Poker?

In [16]:

```
# probability should be close to analytical value of 0.021128

seed(0)
num_trials = 10**5
value = probability_of_rank_signature([1,1,3],num_trials)
print("Probability of three of a kind in Poker is", value)
```

```
Probability of three of a kind in Poker is 0.02073
```

Problem 12 (E): What is probability of a Full House in Poker?

In [17]:

```
# probability should be close to analytical value of 0.001441

seed(0)
num_trials = 10**5
value = probability_of_rank_signature([2,3],num_trials)
print("Probability of full house in Poker is", value)
```

```
Probability of full house in Poker is 0.0015
```

Problem 12 (F): What is probability of Four of a Kind in Poker?

In [18]:

```
# probability should be close to analytical value of 0.000240

seed(0)
num_trials = 10**5
value = probability_of_rank_signature([1,4],num_trials)
print("Probability of four of a kind in Poker is", value)
```

```
Probability of four of a kind in Poker is 0.00022
```

Optional: What is probability of a Straight Flush in Poker?

We will ignore the difference between this and a Royal Flush, so this is essentially calculating the cumulative probability (probability of this hand or better).

You will have to do this at least 10**6 times to get an accurate result.

In [19]:

```
# probability should be close to analytical value of 0.000015 = approximately 1.
539 * 10^(-5)

seed(0)

num_trials = 10**6
```