Question 1:

```
from scipy.special import comb
import math
```

a)

```
# Order does not matter so 40 choose 12
ans = comb(40,12)
print(ans)
```

```
5586853480.0
```

b)

```
# Use the multiplication rule
# Number of men choices * Number of women choices
men = comb(17,6)
women = comb(23,6)
print(men*women)
```

```
1249320072.0
```

c)

```
# The order matters and there are 12 benches and 12 people
permutations = math.factorial(12)
print(permutations)
```

```
479001600
```

d)

```
# There are 12! ways to seat the 12 jurors when rotations do not matter
initial = math.factorial(12)
# We then remove all 12 rotations of jurors using the multiplication rule
permutations = initial/12
print(permutations)
```

```
39916800.0
```

e)

```
# Order matters
# First choose the foreman
foreman = 12
# Then choose the secretary
secretary = 11
# Then choose a vote tabulator
tabulator = 10
# By the multiplication rule
permutations = foreman*secretary*tabulator
print(permutations)
```

```
1320
```

Question 2:

Total: 29 students

A : 9 students

B : 12 students

C : 8 students

You are putting together a committee of 6 students

```
from scipy.special import comb
import math
import numpy as np
```

a)

```
# Probability(3 B and 3 C) = (number of 3 B and 3 C) / total # of choices

b_and_c = comb(8, 3) * comb(12, 3)
answer = b_and_c / comb(29, 6)

print(answer)
```

```
0.025935750073681107
```

b)

```
# The total number of non-A students after 9 pulls is 20 choose 9
no_A = comb(20, 9)
# The total combinations is 29 choose 9
answer = no_A / comb(29, 9)
print(answer)
```

```
0.016770835361540008
```

Question 3

a)

- $A_1 \cup A_2 \cup A_3$
- $A_1 \cap A_2 \cap A_3$
- $A_1 \cap A_2^c \cap A_3^c$
- $(A_1 \cap A_2^c \cap A_3^c) \cup (A_1^c \cap A_2 \cap A_3^c) \cup (A_1^c \cap A_2^c \cap A_3)$

b)

P($A_2 \cap A_3$) = 0.5
P($(A_2 \cup A_3)^c$) = P($A_2^c \cap A_3^c$)
P($A_2^c \cap A_3^c$) = P(A_2^c) + P(A_3^c) - P($A_2^c \cup A_3^c$)
P(A_2^c) = 1- 0.6 = 0.4
P(A_3^c) = 1- 0.7 = 0.3
P($A_2^c \cup A_3^c$) = P($A_2 \cap A_3$) = 0.5
P($A_2^c \cap A_3^c$) = 0.4 + 0.3 - 0.5 = **0.2**

c)

P($A_1 \cup A_2$) = 0.82
P($A_1 \cap A_2 \cap A_3$) = 0.45
P(Reaching the restaurant) = P($A_3 \cup (A_1 \cap A_2)$)
P($A_3 \cup (A_1 \cap A_2$) = P(A_3) + P($A_1 \cap A_2$)- P($A_3 \cap (A_1 \cap A_2)$)
P(A_3) = 0.7
P($A_1 \cap A_2$) = P(A_1) + P(A_2) - P($A_1 \cup A_2$) = 0.8 + 0.6 - 0.82 = 0.58
P($A_3 \cap (A_1 \cap A_2)$) = P($A_3 \cap A_1 \cap A_2$) = 0.45
P(Reaching the restaurant) = 0.7 + 0.58 - 0.45 = **0.83**

Question 4

a)

```
#Probability of selecting 5 numbers between 1 and 39 (order doesn't matter)
select_5 = comb(39, 5)
#Getting one specifc combination of numbers would be 1 in 575757
print("There are " + str(select_5) + " combinations of 5 lottery numbers")
```

```
There are 575757.0 combinations of 5 lottery numbers
```

b)

The sample space Ω = {(A,B)} where A is the set of numbers drawn first and B is the set of numbers drawn second.
|(A,B)| = |A| × |B| |(A.B)| = 575757 × 575757 = 331496123049 ≈ 3.31 × 10^{11}

c)

Suppose my favorite 5 numbers are {8, 12, 27, 16, 31}, the probability of wining the Take5 lottery twice with the same 5 numbers would be 1 in the sample space from part b. Thus the probability would be **3.0166265×10**$^{-12}$ This compares to the given 1 in 331 billion chance as stated by the Harvard Professor.

d)

The odds of winning the morning and evening Take5 lottery with the same 5 numbers would be **1 in 575,757**, since you would treat the numbers you won with in your morning draw, as the set in your evening draw, which compares to the odds of matching all 5 numbers in one draw.

e)

The total number of combinations would be 917 + 916 + 915 + ... + 1, because, using the addition rule matching the kth day with any following day is equal to 918 - k. Therefore, the probability of getting any matching day would be that combination over all possible combinations of drawing 5

```
sum = 0
for x in range(1, 918):
  sum += x
print(sum/575757)
```

```
0.7310427836743626
```

Question 5

Surjective mapping of contracts to contractors

a)

p(n, 1) = 1 for n > 1 because if there is only one set to partition into, we will partition all objects into the same bin, and there is only one way to do that.

b)

p(n, k) = 0 if k > n because we want at least one object in every bin, and if there are more bins than objects.

c)

$k \times p(n-1, k) + p(n-1, k-1)$ can be broken into two parts using the addition rule. We can imagine lining up all of the bins, and starting from the left, either placing he object in the bin or moving to the bin to right until there are no more objects or bins. The left side represents putting an object in the bin, so by the multiplication rule, there are $k \times p(n-1, k)$ combinations. The right side represents moving to the right and placing it in that bin because each bin needs an object. Therefore, by the addition rule $p(n,k) = k \times p(n-1, k) + p(n-1, k-1)$

d)

```
def partition (n, k):
  if k == 1:
    return 1
  elif k > n:
    return 0
  return k * partition(n-1, k) + partition(n-1, k-1)

print(partition(20,9))
```

    12011282644725

e)

```
seconds = partition(20,9) / 1000000
minutes = seconds / 60
hours = minutes / 60
days = hours / 24
print(days)
```

    139.0194750546875

f)

We have a proof that we would not take ORIE 3300, but the proof is too small to fit in this margin.

0s completed at 5:42 PM