```
Math 55 - Spring 2004 - Lecture notes # 13 - March 4 (Thursday)

   Goals for today: Start Chapter 4 (counting and probability)
   Homework: start reading Chapter 4 (sections 4.1, 4.2, 4.3)

   Simple Motivating example for Chapter 4:
      Suppose you are in charge of making up the rules for acceptable
      passwords for customers of a new "dot com". What should the rules
      be to prevent a hacker from being able to try guessing passwords,
      i.e. trying a large number of random passwords in the hopes of
      getting lucky? In other words, how should you make the rules for legal
      passwords so that if a hacker tried this, testing one password
      every, say, 1 microsecond, it would probably take at least a month to
      find one?
ASK&WAIT: suppose you permitted 1 upper case letter passwords?
          How long would it take to try them all?
ASK&WAIT: suppose you permitted 2 upper case letter passwords?
ASK&WAIT: suppose you permitted 6 upper case letter passwords?
ASK&WAIT: suppose you permitted 8 letter passwords?
ASK&WAIT: suppose you permitted 8 character passwords,
      with at least one nonletter?
ASK&WAIT What is the chance of finding one in a month of trying random guesses?
      is this unlikely enough?

  Counting Principles
  1) The Sum Rule:
  EX: If you have to do one project for a class, and are given one list with
      2 projects and another with 3 different projects, how many different
      projects do you have to choose from? 2+3 = 5

  The Sum Rule (formally): Suppose we have two tasks to do, T1 and T2. Let
     S1 be the set of n1 ways to do task 1, and S2 the set of n2 ways to do
     task 2, where S1 and S2 disjoint. The set of ways to do either
     T1 or T2 is S1 U S2.  The number of ways to do either T1 or T2 is
     |S1 U S2| = |S1| + |S2| = n1 + n2

  2) The Product Rule:
  EX: If you have to do two projects for a class, the first one chosen from a
      list of 2 projects, and the second one chosen from a list of 3 project,
      how many different pairs of projects could you turn in?
      S1={p1,p2}, S2={pa,pb,pc}
      pairs={(p1,pa),(p1,pb),(p1,pc),(p2,pa),(p2,pb),(p2,pc)} = S1 x S2
```

```
     2*3 = 6 different pairs

  The Product Rule (formally) Suppose we have two tasks to do, T1 and T2,
     with S1, n1, S2, n2 as above. The set of ways to do both T1 and T2 is
     S1 x S2. The number of ways to do both S1 and S2 is
     |S1 x S2| = |S1| * |S2| = n1 * n2
     (remember S1 x S2 is the set of all pairs of entries {(x1,x2), xi in Si}

  3) The Extended Product Rule: If S1 is the set of n1 ways to do T1,
     S2 the set of n2 ways to do T2, ... , Sm the set of nm ways to do Tm,
     then the set of ways to do T1,T2,...,Tm is S1xS2x...xSm, which has
     n1*n2*...*nm elements

ASK&WAIT: How many bits strings of length 9 are there?
ASK&WAIT: How many different license plates are there if all consist
          of three letters following by 3 numbers?
ASK&WAIT: How many different computer passwords are there if they may be
          8 characters, upper case letters only?
ASK&WAIT: How many different computer passwords are there if they may be
          6-8 characters long, upper or lower case letters, digits?
ASK&WAIT: What if there must be at least one letter and one number?

ASK&WAIT: How many functions f:X->Y are there, if X has m elements and Y has n?

ASK&WAIT: How many one-to-one functions are there from S to T?

ASK&WAIT: How many ways can you shuffle a deck of 52 cards?

    EX: How many ways can a class of 100 students be divided in 2-student
        teams?
        2 students {s1,s2} -> 1 way
        4 students {s1,s2,s3,s4} -> 3 ways
             {(s1,s2),(s3,s4)}, {(s1,s3),(s2,s4)}, {(s1,s4),(s2,s3)}
        How do we get a simple formula for any even n?
          Suppose there are are P(n-2) pairings of n-2 students,
          whose names are 1, 2, ... , n-2; now add students n-1 and n
          What pairings are possible?
                Take student n, and choose any other student m to make
                   the pair.  that leaves n-2 students, with P(n-2)
                   possible pairings.
                 m can take on n-1 values, so there are (n-1)*P(n-2)
                   possible pairings.
```

```
          Result: recurrence P(n) = (n-1)*P(n-2) , with P(2)=1
          Are we sure we have counted every possibility exactly once?
          Use induction: assume P(n-2) is correct
             In construction, get (n-1) groups of P(n-2) pairings, where
                n is paired with a different m in each group. So
                no pairing appears in more than one group. And no pairing
                can appear twice in one group because all P(n-2) groupings
                of n-2 students are different, by induction. And each
                pairing has to appear in one group, depending on parter of n.
ASK&WAIT: What is a closed form formula for P(n)?
        For n=100: P(100) = 99*97*95*...*3 \approx 3e78
                           n  P_n
                           2  1
                           4  3
                           6  15
                           8  105
                          10  945
                          20  6.5e+08
                          40  3.2e+23
                          60  2.9e+40
                         100  2.7e+78
                         150  6.1e+130
                         350  2.3e+369
      (number of atoms in universe once thought to be about 1e80)

  4) Inclusion-Exclusion Principle:
  EX: How many 8-bit strings either start with 1 or end with 00?
      S1 = {1xxxxxxx, x= any bit}, S2 = {xxxxxx00, x=any bit}
      We want |S1 U S2|. But S1 and S2 overlap: S1 inter S2 = {1xxxxx00}
      So we count |S1| = 2^7, |S2| = 2^6. But |S1|+|S2|>|S1 U S2| because
      S1 inter S2 has been counted twice, so we subtract it:
      |S1 U S2| = |S1| + |S2| - |S1 inter S2| = 2^7 + 2^6 - 2^5 = 160
  The Inclusion-Exclusion Principle (formally) Suppose we have two tasks to do,
     T1 and T2, with S1, n1, S2, n2 as above, except S1 and S2 may intersect.
     The set of ways to do both T1 and T2 is S1 U S2. The number of ways to do
     both S1 and S2
     |S1 U S2| = |S1| + |S2| - |S1 inter S2|
  EX: How many <= 3 decimal digit numbers are divisible by 3 or by 4?
  Inclusion-Exclusion with 3 tasks (see Question 1.5.34)
      Suppose you have 3 tasks, in sets S1, S2, S3, which might overlap.
      Then | S1 U S2 U S3 | = |S1| + |S2| + |S3|
                             - |S1 inter S2| - |S2 inter S3| - |S1 inter S3|
```

+ |S1 inter S2 inter S3 |

EX: How many <= 3 decimal digit numbers are divisible by 3, 4 or 5?