CS 542, Homework #1

Write the following functions (in which you don't have to worry about overflow).

Also, write a main that calls these functions. The purpose of main is to show you that your functions probably work. You decide the details of your main; my interest is in your functions, I don't really care how you decide to write your main.

Please use the exact function headers that I specify.

unsigned **factorial**(unsigned n);

Return the factorial function of n. For example, factorial(4) should return 24

unsigned **comb**(unsigned n, unsigned k);

Return the number of combinations of n things taken k at a time.

For example, comb­(5, 2) should return 10 because there are 10 combinations of 2 things taken from ABCDE:

AB

AC

AD­

AE

BC

BD

BE

CD

CE

DE

(AB and BA are considered to be the same combination.)

unsigned **perm**(unsigned n, unsigned k);

Return the number of permutations of n things taken k at a time.

For example, perm(5, 2) should return 20 because there are 20 permutations of 2 things taken from ABCDE:

AB

AC

AD

AE

BA

BC

BD

BE

CA

CB

CD

CE

DA

DB

DC

DE

EA

EB

EC

ED

(AB and BA are considered to be the different permutations.)

unsigned **combABC**(unsigned n);

combABC outputs each combination of n letters, where each letter is A, B, or C. For example, combABC(4) would output 15 lines:

AAAA

AAAB // or it could just as well output ABAA, since that's the same combination

AAAC

AABB

AABC

AACC

ABBB

ABBC

ABCC

ACCC

BBBB

BBBC

BBCC

BCCC

CCCC

combABC should also return the number of combinations; it should be the number of combinations by computation, not just counting how many lines of output it produces.

unsigned **permABC**(unsigned n);

permABC outputs each permutation of n letters, where each letter is A, B, or C. For example, permABC(2) would output 9 lines:

AA

AB

AC

BA // different permutation from AB, even though they're the same combination

BB

BC

CA

CB

CC

permABC should also return the number of permutations; it should be the number of permutations by computation, not just counting how many lines of output it produces.

unsigned **f**(unsigned a, unsigned b, unsigned c);

Compute and return the number of strings of a A's, b B's, and c C's. For example, f(1, 2, 1) returns 12 because there are 12 strings made up of 1 A, 2 B's, and 1 C:

ABBC

ABCB

ACBB

BABC

BACB

BBAC

BBCA

BCAB

BCBA

CABB

CBAB

CBBA