BRONZE PROBLEMS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Three problems numbered 11 through 13

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem 11: Cruel Math Teacher, I [Traditional, 2009]

Bessie has returned to 8th grade in order to finish her diploma.

Her cruel math teacher wants the students to calculate "powers of

integers". An integer power is the resultant integer when some

number N (1 <= N <= 2,000,000,000) is multiplied by itself over and

over P times (1 <= P <= 100,000).

By way of example, 2 to the power 3 = 2 \* 2 \* 2 (three times) = 8.

Similarly, 123456 to the power 88 = 123456 \* 123456 \* ... \* 123456 (88

times) =

1129987770413559019467963153621658978635389622595924947762339599136126

3387265547320084192414348663697499847610072677686227073640285420809119

1376617325522768826696494392126983220396307144829544079751988205731569

1498433718478969549886325738202371569900214092289842856905719188890170

0772424218248094640290736200969188059104939824466416330655204270246371

3699112106518584413775333247720509274637795508338904731884172716714194

40898407102819460020873199616

when printed 70 digits per line.

Write a program to calculate the Pth power of an integer N. The

answer is guaranteed to be no longer than 15,000 digits. Print your

answer 70 digits per line (except the last line which might be

shorter). Do not print leading zeroes, of course.

PROBLEM NAME: cruel1

INPUT FORMAT:

\* Line 1: Two space-separated integers: N and P

SAMPLE INPUT (file cruel1.in):

2 15

INPUT DETAILS:

Calculate 2 to the 15th power.

OUTPUT FORMAT:

\* Lines 1..?: A single integer that is the result of the calculation.

Print 70 digits per line except potentially for the last line,

which might be shorter.

SAMPLE OUTPUT (file cruel1.out):

32768

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem 12: Cruel Math Teacher, II [Traditional, 2009]

As if powers of numbers were not cruel enough, Bessie's cruel math

teacher has created yet another cruel assignment: Find the 'root'

or 'zero' of a polynomial. All of these polynomials have a highest

degree D (1 <= D <= 11) that is odd and have but a single solution

in the range -1,000,000 <= X <= 1,000,000; for that solution, the

polynomial's value when evaluated for X is very close or equal to

to 0 in computer math.

Given a polynomial with real number coefficients (-500 <= coef\_i

<= 500), find a value of X that is within 0.0005 of the value of X

that will yield 0 when the polynomial is evaluated. Multiply that

value of X by 1,000 and print it as an (unrounded) integer.

By way of example, consider the cubic polynomial problem 1.5\*x\*x\*x

- 10 = 0. Astute algebra students will quickly recognize a solution

for this as x\*x\*x = 100/15 = 20/3 = 6.66666. To five decimal places,

the exactly solution is 1.88207. For this task, the proper output

would be 1882.

The polynomial is expressed as the sum from i=0..D of coef\_i\*x^i

(where x^i means x to the i-th power).

No answer will require more than six significant digits and each

answer will be small enough that it is able to be incremented by

0.0001 in the 'double' precision floating point datatype without

losing lots of precision.

HINT: Find a strategy to narrow the search space each time you

choose a new X value as a guess.

NOTE: Your first 50 submissions will report results of more test cases

than just the first one.

PROBLEM NAME: cruel2

INPUT FORMAT:

\* Line 1: A single integer: D

\* Lines 2..D+2: Line i+2 contains a single real number: coef\_i

SAMPLE INPUT (file cruel2.in):

3

-10.0

0.0

0.0

1.50

INPUT DETAILS:

[As in the example from the text.]

OUTPUT FORMAT:

\* Line 1: A single integer that is the truncated product of 1,000 and

the X value that is closest to the X value that causes the

polynomial to evaluate to 0

SAMPLE OUTPUT (file cruel2.out):

1882

OUTPUT DETAILS:

As in the text.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem 13: Coggle [Traditional, 2009]

Like everyone else on their vacation, the cows play the cow version of the

word game called Boggle: Coggle. It's a similar game where 25

letter dice are rolled into a matrix like this one:

Z C C D X

K Q M N B

U O W Z Y

F C O I J

P A Q Z T

Words are made (and thus points scored) by starting at some letter

and proceeding to one of its (as many as) eight neighbors, etc.

until the successive letters spell out a word from a dictionary

In the matrix above, the lower 'C' can be used to form the words

"CAP", "COW", and "COOK" (but not "COD" or "PACK"). The complete

list of dictionary words for the above square is: "CAP", "COOK",

"COW", "OWN", "WIT", "WOO", "ZOO", and "ZOOM".

Your program should read the dictionary from file dict.txt (which is

alphabetized and has fewer than 25,000 words; each word is no longer

than 20 characters). The actual dictionary contents can be inspected

at <http://ace.delos.com/usaco/dict.txt>.

Help Bessie see how good she can do. Read in five rows of five

letters that represent the dice and see how many words from the

dictionary can be formed. Don't use any given die's letter twice in

the same word.

Count the number of words that can be formed (a number that might

well be smaller than the number of ways a word a can be formed since

words might be formed in more than one way).

PROBLEM NAME: coggle

INPUT FORMAT:

\* Lines 1..5: Line i contains five space-separated upper-case letters

that are row i

SAMPLE INPUT (file coggle.in):

Z C C D X

K Q M N B

U O W Z Y

F C O I J

P A Q Z T

OUTPUT FORMAT:

\* Line 1: A single integer that is the number of words in the

dictionary that can be formed using the described rules

SAMPLE OUTPUT (file coggle.out):

8

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*