

A General Summary of COMCON Programming Contest

(A sample contest follows this description)

Contests are held about five times per school year, usually about once a month. The **contests are free; there is no fee** of any kind. While coming to the contest site is encouraged, teams **may participate over the internet: there is NO TRAVEL required.**

The contest consists of four programs. As many of these as possible are to be solved by a team (up to four people) using one computer within a two hour limit. **Schools can compete alone or against other schools;** to compete against other schools successful program solutions must be emailed (see address below).

Programs are worth 100 points each. There is a time penalty of one point for every five minutes of elapsed time before a program is solved. A “bad run”, that is a program that does not give the correct output (or that crashes or fails to run properly) is an additional ten point penalty. See the “additional scoring” notes below.

A submission is made with the team’s flash drive or disk, indicating the team name, program number, and the language used. (If a school has more than one team, the designation “A team”, “B team”, etc. is also included). The judges will fill in the time the program was submitted.

The program submission is named PROGRAM1, PROGRAM2, etc. and has an extension indicating the language in which it is written: .CPP, .perl, .java, etc. The programs may be submitted in any order. Because COMCONs are fundamentally contests of speed, a good strategy is to submit the easiest program first.

The programs are tested by running the program, which opens the test data file on the C: drive or some other location (C:\PROGRAM1.DAT, C:\PROGRAM2.DAT, etc.). Programs must complete execution and terminate within 60 seconds. Judges will run the program using test data, determine successful runs by visual inspection of the output, and will report the results as soon as possible.

If a program does not work with the test data, a bad run will be noted, and a comment will be given to the team indicating the general nature of the problem.

A scoreboard will indicate what teams have solved which programs, and the relative standings of the meet.

The team with the highest score wins, with the exception that a team that solves more total programs than another wins, regardless of time.

The decisions of the judges are final.

Questions? Please email Rich Lamb: Rlamb@cranbrook.edu

Additional Scoring Notes

The contest runs for two hours. There are four problems in each contest. A team may have one or more students to a maximum of four. Regardless of the number of team numbers, a team may only

use one computer. Programs may be submitted in any order. Because the contest is largely one of speed, it is a good strategy to submit the easiest program first.

Here is an example of scoring: If I begin a contest at 4:00 and turn in my first problem say at 4:17, that means I have used 3 five-minute time blocks (4:00 to 4:05, 4:05 to 4:10, and the next two minutes, which also counts as a five-minute block). Thus, I have three points of time against me, so I score a 97 ($100 - 3$) for the first problem.

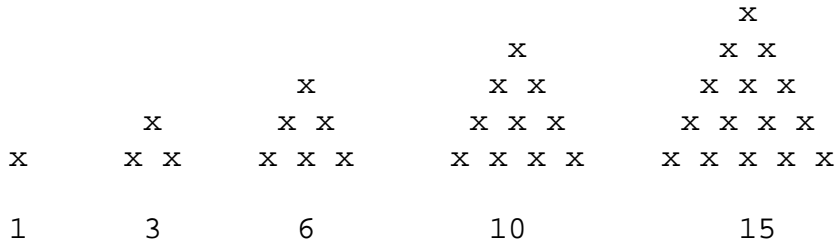
Now let's say that my second correct problem comes in at 4:31. The clock has been ticking since 4:00, so I have 7 points against me, and I score a 93 ($100 - 7$) for this second problem.

Each time I have a "bad run" (the program crashes, takes more than one minute to execute, or gives the wrong output), I receive a ten point penalty **whether or not I subsequently successfully complete that problem.** (Penalties hurt!)

So, for example, let's say I submitted program 3 at 4:17, then program 2 right after, but it did not work, and then I submitted program 1 at 4:31. My score would be $97 + 93 - 10 = 180$. Then at 5:01 I correctly submit program 2, and make no more submissions. My final score is $97 + 93 + 89 - 10 = 269$.

“Triangle Numbers”
COMCON
9 November 2005
Program 1

Triangle numbers are those from which a triangular shaped pile may be made from that number of objects:



The sequence is 1, 3, 6, 10, 15, 21... and is infinite. In the 5 groupings above, the *number* of the groups are 1, 2, 3, 4, and 5 (that is, the first, second, third, fourth and fifth triangle groups) and the *values* of the groups are 1, 3, 6, 10, and 15, respectively.

The input file consists of numbers, which either is preceded by an “N” for number, or a “V” for value. For the “N” number, you are to print out the corresponding value; for the “V” number, you are to print the corresponding number. It may be that some “V” numbers are not correct, that is, they are not triangle numbers. In this case, you are to print “incorrect number”. The flag of “END” will terminate the data. The data file will not exceed 1,000,000 for a value.

SAMPLE INPUT (C:\PROGRAM1.DAT)

```

N3
V6
V200028
N1983
N51
V1000
END

```

SAMPLE OUTPUT

```

6
3
632
1967136
1326
INCORRECT NUMBER

```

“Bored Jailer”
COMCON
9 November 2005
Program 2

This is an adaptation of an old puzzle. A jailer is in charge of N cells, $1 \leq N \leq 1000$. One day to relieve boredom, the jailer decides to go to every X th cell, with wrap-around if necessary, and to visit T cells in total. When arriving at a cell, the lock is toggled, that is, if it is locked, it is now unlocked, and vice-versa. The cells are all initially locked. The program will then print a range of cells given by the data and printed the number, in order, of each unlocked cell.

The data has values of N , X , and T in that order, followed by a pair of numbers that represent the range of cells to print. The first of the pair will always be less than or equal to the second. X will be less than N , and T will be no greater than one million (the jailer is *very* bored).

SAMPLE INPUT (C:\PROGRAM2.DAT)

```
10
4
7
3
8
```

Here the jailer has 10 cells, and will visit every 4th cell for seven times. Hence, the cells visited are 1, 5, 9, 3, 7, 1, and 5, in that order.

SAMPLE OUTPUT

```
CELL 3 IS UNLOCKED
CELL 7 IS UNLOCKED
```

**“Squares & Cubes”
COMCON
9 November 2005
PROGRAM 3**

This is strange and yet weird problem, as it has no input file, only output.

There is one three-digit number and one four-digit number that are both perfect squares and perfect cubes. Find and print these numbers.

Then, to help alleviate your boredom, print the next highest number that has this property.

Thus, the program will output three numbers. A reminder your program must execute in one minute or less.

SAMPLE INPUT (none)

SAMPLE OUTPUT (but not correct! ☺)

111
2222
10000

“Some Assembly Required”
COMCON
9 November 2005
Program 4

The COMCON assembler consists of an accumulator, a temporary storage area, and 10 registers, R0 through R9. The machine has the following instructions, where the operand is either an identifier or a storage location.

L	load the operand into the accumulator
A	add the operand to the contents of the accumulator
S	subtract the operand from the contents of the accumulator
M	multiply the contents of the accumulator by the operand
D	divide the contents of the accumulator by the operand
N	negate the contents of the accumulator
T	store the contents of the accumulator in the operand location

An arithmetic operation replaces the contents of the register with the expression result. Temporary storage locations are allocated by the assembler for an operand of the form “Rn” where n is a single digit.

The input file consists of a (syntactically correct) postfix expression. Expression operands are single letters and operators are the normal arithmetic operators (+, -, *, /). Output must be assembly language code that meets the following requirements:

1. One instruction per line with the instruction mnemonic separated from the operand (if any) by one blank.
2. One blank line must separate the assembly code for successive expressions.
3. The original order of the operands must be preserved in the assembly code.
4. Assembly code must be generated for each operator as soon as it is encountered.
5. As few registers as possible should be used (given the above restrictions).
6. For each operator in the expression, the minimum number of instructions must be generated (given the above restrictions).

Sample input
(C:\PROGRAM4.DAT

Sample output

AB+CD+-

```
L A
A B
T R0
L C
A D
N
A R0
```

(note only one register is used)

TEST DATA - November 2005

Triangle numbers PROGRAM1.DAT

V1234567
N1571
V7112106
N700
N2145
N4471
V10000000
N17906
END

Bored Jailer PROGRAM2.DAT

170
4
253
8
15

Squares & Cubes PROGRAM 3

<no data file>

Some Assembly Required PROGRAM4.DAT

AB+CD+EF++GH+++

TEST OUTPUT

Triangle Numbers - Program 1

INCORRECT NUMBER
1234806
3771
245350
2301585
9997156
INCORRECT NUMBER
160321371

Bored Jailer - Program 2

cell 9 is unlocked
cell 11 is unlocked
cell 13 is unlocked
cell 15 is unlocked

Squares and Cubes - Program 3

729
4096
15625

Some Assembly Required - Program 4

Note, that judges should examine output if it does not exactly match what is given here. There are a number of possibilities that can be correct.

L A
A B
T R0
L C
A D
T R1
L E
A F
A R1
T R1
L G
A H
A R1
A R0