

TAMS Tournament Programming – Team Packet

TAMS Computer Science

November 15, 2011

Rules

Welcome to the TAMS Tournament programming event! Inside this packet are the problems for the event; do not open until you are told to do so. Outside, you should have this page and the DOMjudge team manual, which covers solution submission and clarification requests.

Here are a few event rules:

- Keep reasonably quiet while the event is in progress
- Web access is not permitted; you should only connect to the event network (which does not have Internet access)
- Only one computer is allowed per team
- Problems may be answered in any order, and are equally weighted
- In the event of a tie, DOMjudge will select the team with the lowest total time as the winner (see TAMS Tournament website for details)
- You should work as a team to solve the problems correctly
- The only reference material that can be used during the competition is API
- Solutions are submitted as source code, which will be compiled on the server
- Additional questions should be asked as a clarification request

You should proceed to connect to the wireless network *ttprogram*. Then, open a web browser and navigate to *http://ttprogram/*. Login with your team name and team password that should have been given to you. When the contest starts, you will be able to submit problems, send clarification requests, view scores, etc.

Each problem has a full name and a short name. In the packet, in the section header, the short name is inside parentheses. Your source files should be named in accordance with the short name (this isn't too important as DOMjudge will rename your files anyway; but if you name your files correctly, you can submit faster and the problem can be autodetected). All input is done through standard input and all output is done through standard output (from and to console).

Binomial Expansion (binomial)

Binomial Expansion is a mathematical way to expand an expression, such as $(x+y)^2$, to $x^2+2xy+y^2$. Oftentimes the most difficult part is finding the coefficient to each term. To find each coefficient, the Combinations can be used. The formula for combinations is $\frac{n!}{k!(n-k)!}$, where n is the term number (starting from 0) and k is the exponent.

Input (from standard input)

The first line of input will be a number n , designating the number of expansions that are in the set. The next $n - 1$ lines will be the expressions that need to be expanded, followed by an empty line. There will never be a coefficient to the terms in the input line, but you must output the expanded equation with coefficients. The first part will be a variable and the second part will be an integer. The integer will be between 0 and 5 inclusive, and the exponent will be between 0 and 10 inclusive.

```
3
(x+1)^2
(w+5)^3
(z+2)^7
```

Output (to standard output)

If the exponent of a term is 1, then do not display the exponent. If the exponent is 0, then do not display the variable. If the value of the term is 0, do not display the term. For example, $3x^1$ should be printed as $3x$, $3x^0$ as 3, and $x + 0$ as x .

```
x^2+2x+1
w^3+15w^2+75w+125
z^7+14z^6+84z^5+280z^4+560z^3+672z^2+448z+128
```

Currency exchanging (currency)

At the Ferengi currency exchange at the Tower of Commerce on Ferenginar, you can only exchange 10,000 units of each currency. In return, you get 10,000 units of another currency. If the two currencies have different values per unit, then any additional costs are paid in latinum (the standard currency). For example, if you exchange 10,000 sinics for darseks, you get 10,000 darseks but additionally are given 12 bars of latinum. On the other hand, if you exchange 10,000 darseks for doraks, you receive 10,000 doraks but additionally are given -15 bars of latinum (you have to pay 15 bars). There is a restriction on currency exchanging: only exchanging between currencies for which rates are posted can be completed, and rates are not posted between every pair of currencies. So, if you exchange from sinics to darseks, you might not be able to directly exchange back from darseks to sinics.

You want to take advantage of the Ferengi currency exchange. Sometimes, the rates are set such that if you complete a certain set of exchanges, you both receive latinum and return to your original currency. For example, if conversion from sinics to darseks yields 12 bars of latinum, conversion from darseks to doraks yields -15 bars of latinum, and conversion from doraks to sinics yields 4 bars of latinum, each time you complete the set of conversions you receive 1 latinum. There are no restrictions on the volume of exchanges, so you can continue earning bars of latinum until you get too rich (which is not possible: "greed is eternal" according to the tenth rule of acquisition) or the rates are updated.

Write a program that can test whether or not such a set of conversions exists, and output true or false accordingly.

Input (from standard input)

The first line is the number of rates. Each line after contains one rate, which consists of the names of two currencies followed by the number of latinums received for the exchange. These are space-separated.

```
3
sinics darseks 12
darseks doraks -15
doraks sinics 4
```

Output (to standard output)

Output "true" if the posted rates can be exploited to earn infinite sums of money, or "false" if you can't get rich.

```
true
```

Maximize Equation (maximize)

Justin Bieber has hired you to draw maximize several equations as he wants to improve profits from his music. Each equation has up to three variables (denoted as 'x', 'y', and 'z'), and you are given the domain of each variable in the equation (the domain will only include integers; however, there may be floating point constants in the equations; furthermore, the range of the domain will not exceed 50). Valid operations are addition ('+'), subtraction ('-'), multiplication ('*'), and division ('/'). Multiplication will always be explicitly shown (i.e., you will not see $.25x$, only $.25 * x$). Valid functions are sine ('sin'), cosine ('cos'), base two log ('log'), and tams ('tams'; $tams(x, y) = x! + (x + y)! = x(x - 1)(x - 2)...(2)(1) + (x + y)(x + y - 1)...(2)(1)$). Functions will always use parentheses and, if there are more than two operands (as in the tams function), commas. Also, to avoid confusion with subtraction, negative numbers will always be representing as zero minus a positive number. All tokens will be separated by spaces in the equation.

Input (from standard input)

The first line of the input file will contain the number of variables N in the equation Bieber gives you. The next N lines contain the variable name and the domain for that variable (variable names will always be 'x', 'y', and 'z', in that order, but it will anyway be included in the input file). The next line will contain the equation, with tokens separated by spaces, to be maximized in the domain.

```
2
x:1 to 9
y:-5 to -3
( 0 - 1 ) * ( x - 2 ) * ( x - 2 ) + 1.01 * sin ( x ) + y * log ( x ) + tams ( 0 , 1 )
```

Output (to standard output)

The first N lines should contain the variables (in same order as in input file, which, again, will be 'x', then 'y', then 'z') and their values that maximize the equation. The last line should contain the maximum value of the function. The maximum value should always include exactly five digits after the decimal point.

```
x=1
y=-5
1.84989
```

Quadratic Solutions (quadsol)

You were recently hired by MathWorks to design an algorithm that can automatically solve quadratic equations. You will be given equations in the form $0 = ax^2 + bx + c$. Then, you should use the quadratic formula, $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ (the variables a , b , and c in the formula are the same as in the equation). Your program should be able to handle both real and non-real complex solutions.

Input (from standard input)

The first line in the input will be the value of a , the second line the value of b , and the third line the value of c . The example corresponds with the equation $0 = 2x^2 + 4x + 5$.

```
2
4
5
```

Output (to standard output)

The output is divided into two sections of two lines each. The first line in each section is the real part, and the second line is the imaginary part (in $a + bi$, a is on the first line and b is on the second line). Each part should contain exactly four digits after the decimal place. The solution with the smaller real part should be written first. If the real part is the same, the solution with the smaller imaginary part should be written first.

```
-1.0000
-1.2247
-1.0000
1.2247
```

Remembering Questions (remq)

Its two hours before TAMS Tournament and we need to organize the questions. However, there was a mess up and the problems we wrote for this competition got lost among the other problems we had already written. Fortunately, we remembered that each of the problems we made for the tournament was a palindrome, and each of the other problems was not. Help us by writing a program that can sort through a list of problems and generate the list that will be used today. Case matters.

Input (from standard input)

Line 1 will contain a number n that is the number of problems are on the list. Line n+1 will contain a String that is the name of the problem.

```
5
Pie
OMG what?
PieeiP
Ladyydal
Happy Birthday!!yadhtriB yppaH
```

Output (to standard output)

Several lines that contain a list of the problems that we need.

```
PieeiP
Happy Birthday!!yadhtriB yppaH
```

Schedule (schedule)

Darren needs to figure out his schedule for his classes next semester. He has already created a program to output a bunch of schedules he could use. The output contains a list of classes and the hour that each class takes place at (all classes are one hour long at his school). He has to follow the entire schedule because there are requirements for what he has to take. Unfortunately, there are too many schedules. This normally wouldn't be a big problem, as Darren could just take the first schedule, but almost all the schedules are infeasible with overlapping class hours. Help Darren by writing a program to examine a schedule and determine whether any of the classes overlap.

Input (from standard input)

The first line will be the number of classes in the schedule. Each line after will contain a classname and the hour that the class occurs at. They will be space separated, and there will be no spaces in the class name. Also, the hour will be between 0 and 23.

```
5
Physics 5
English 7
AdvancedAddition 1
History 6
Geometry 5
```

Output (to standard output)

Output "OK" if the schedule looks all right or "BAD" if the schedule contains any overlaps. In the sample, Geometry and Physics collide (both occur at hour 5), so the schedule is bad.

```
BAD
```

Sequence Distances (sequence)

Dr. Carsch is attempting to determine the relatedness between two unicorn species. To do this, he has sequenced the genomes of the two species; now, he wants you to compare them and number of insertions, deletions, and substitutions that have taken place. There are an infinite number of possible sequences of insertions, deletions, and substitutions: for example, you may begin with TT, insert A to get TTA, then delete A to get back to TT. However, the minimum number of insertions, deletions, and substitutions is the most likely, and this is what Dr. Carsch wishes you to find.

Input (from standard input)

The input file will consist of two lines. Each line will contain one DNA sequence, with each character representing one base pair. Valid base pairs are 'A', 'C', 'G', and 'T' (you will not be given any invalid base pairs). Each sequence will consist of no more than 2000 base pairs.

```
TTATGC
ATTCTC
```

Output (to standard output)

Output the determined minimum number of insertions, deletions, and substitutions.

```
3
```