High School Programming Contest at UTD 4-1-17

How-to-solve

A: Darts

This problem is straightforward. Just add the values for Jane and Joe, 1, 2, or 3 per line, stopping when one of them reaches, or exceeds 301, or when the input is terminated by "End of game"

Notice the periods at the end of the lines of output.

B: Expressions

The *least significant digit* (LSD) is the lowest order digit (in the number 173, the 3 is the LSD).
You should know the theorems of number theory, that:

$$(a+b)\,mod\,c = ((a\,mod\,c) + (b\,mod\,c))\,mod\,c$$

$$(a*b)\,mod\,c = ((a\,mod\,c) * (b\,mod\,c))\,mod\,c$$

Therefore we only need work with the least significant digits of the input numbers.

1. Strip each number of all but its LSD.
2. Do the arithmetic, taking the mod after each operation
3. Take the required mod at the end

For example, for the problem 102345*20646*13*717111 radix 8 do:

5*6 mod 8 = 6
then 6*3 mod 8 = 2
then 2*1 mod 8 = 2

C: Hugs and Kises

There are multiple ways to solve this problem, one or more using Dynamic Programming. We will just look at two simple approaches

The maximum number of problems (20) and the maximum grid size (20x20) makes a brute force approach feasible.

If we are searching for 'x' characters, iterate over the grid and, for each character that is equal to 'x', say at position p,q, calculate the area of largest rectangle containing 'x' characters where p,q is the top left corner of that rectangle.

Here is another approach:

Let $sum[row][col]$ be the summation of all elements from the top left corner of the matrix $(0,0)$ up to the (row, col) position. To count the summation of elements between position (x_1, y_1) and (x_2, y_2) where $x_1 < x_2$ and $y_1 < y_2$, calculate

$$sum[x_2][y_2] - sum[x_1-1][y_1-1]$$

To check whether a given range (x_1, y_1) up to (x_2, y_2) is a rectangle full of '1', check whether

$$sum[x_2][y_2] - sum[x_1 - 1][y_1 - 1] = (x_2 - x_1 + 1) * (y_2 - y_1 + 1)$$

D: Pairs

The straightforward solution:

Consider the problem 1,000,000,000, which was given as the largest problem that could occur. That probably means it will be present in the test data.

The two values that must sum to this number must have 9 and 8 digits. Why? The smallest value you can get with 9d + 8d is 100,000,000 + 00,000,000 = 100,000,000 and the largest is 999,999,999 + 99,999,999 = 1,099,999,998. The problem number is within this range. It seems a good idea to construct a table in the program of these ranges:

2d + 1d:	10	108
3d + 2d:	100	1,098
4d + 3d:	1,000	10,998
5d + 4d:	10,000	109,998
6d + 5d:	100,000	1,099,998
7d + 6d:	1,000,000	10,999,998
8d + 7d:	10,000,000	109,999,998
9d + 8d:	100,000,000	1,099,999,998
10d + 9d:	1,000,000,000	10,999,999,998

etc.

Given a problem number, first search this table to get the one or two ranges into which that value might fall.

Take 1,000,000,000.
It falls into the 9d+8d and the 10d+9d ranges.

Consider the 10d+9d case. The only possible solution is 1,000,000,000 + 000,000,000.

Consider the 9d + 8d range where the last digit is cancelled:

```
 abcdefghk
+ abcdefgh
----------
1000000000
```

a must be 9.
On the LS digit, k+h = 0 or k+h = 10.

If k+h = 0, then g=f=e=d=b=c=a=0, but a=9.

If k+h=10, then there are several cases:

```
 k=9, h=1
 k=8, h=2
 k=7, h=3
```

```
k=6, h=4
k=5, h=5
k=6, h=4
k=7, h=3
k=8, h=2
k=9, h=1
```

Consider the first case:

```
  9bcdefg19
   9bcdefg1
  ---------
 1000000000
```

Then g=8, and the rest follows:

```
  918181819
   91818181
  ---------
 1010000000  Not a valid solution.
```

The second case proceeds similarly:

```
  927272728
   92727272
  ---------
 1020000000
```

From these examples, we can begin to build a strategy:
1. Find the range(s) for the solution from the table
2. Construct a recursive function
`solve(String n1, String n2, String target)`
One of the top level calls to solve this problem would be:
`solve("abcdefg19", "abcdefg1", "1000000000")`
It would call itself as follows:
`solve("abcdefg1", "abcdefg", "100000000")`
and then:
`solve("abcdef8", "abcdef", "10000000")`
etc.

A global list of solutions could be kept so that the function could store the solutions as they are found. Alternatively, solve could return a string containing the solution of the empty string if there is no solution.

The smart solution to the pairs problem:

We have: $z = x + y$, where y is formed from x by removing 1 digit: $49939 = 45369 + 4569$, for example.

Express $z = a.b = p.n.q + p.q$, where a, b, p, q are strings of digits and n is the single digit of x that is removed. The dot means concatenation and the plus sign means an arithmetic add of the two numbers represented by the Strings it is applied to.

$|b| = |q|$ (the length of b equals the length of q.)

In the above example, $a = 499,\ b = 39,\ p = 45,\ n = 3$, and $q = 69$.

If n is in the 10^k position ($k = 2$ in the above) then we can write:

$$a * 10^k + b = p * 10^{k+1} + n * 10^k + q + p * 10^k + q = 11p * 10^k + n * 10^k + 2q$$

We can use this equation to solve the problem.

Consider $z = 302$. If there are m digits in z, then there must be m or $m - 1$ digits in x. Since the most significant digit of z is > 1, there are 3 digits in x. So we have three cases to consider: removing the $k = 0,\ k = 1,\ k = 2$ digits in x.

$k = 0$: (removing the units digit) then $q = 0$ and the equation simplifies to

$$302 = 11p + n$$

Recall that n is a single digit.

The only solution is $p = 27$ and $n = 5$. So one answer to the problem is:

$$302 = 275 + 27$$

$k = 1$: Now q and n both have 1 digit and $302 = 110p + 10n + 2q$
Examining $b = 2$, Clearly $2q = 2$ or $2q = 12$ and q is either 1 or 6.

$q = 1:\ 300 = 110p + 10n$, or $30 = 11p + n$, giving $p = 2$ and $n = 8$ and the second solution:

$$302 = 281 + 21$$

$q = 6:\ 290 = 110p + 10n$, or $29 = 11p + n$, giving $p = 2$ and $n = 7$ and a third solution:

$$302 = 276 + 26$$

$k = 2$: Now q has 2 digits and $302 = 1100p + 100n + 2q$
Clearly $p = 0$, giving $302 = 100n + 2q$. We can see by examining $b = 02$ and remembering that q has 2 digits, $q = 01$, $q = 51$ are solutions (since $2q$ must be 002 or 102)

$q = 01:\ 300 = 100n$, so $n = 3$ giving a forth solution:

$$302 = 301 + 01$$

$q = 51:\ 200 = 100n$, so $n = 2$ giving a fifth solution:

$$302 = 251 + 51$$

How many solutions can there be for an m digit input number? $2m - 1$, since there can be no more than 2 possible values for q at each stage where $k > 0$, and only one value for $k = 0$.

This solution runs in time proportional to the number of solutions, which is $O(m)$, or $O(log\, g)$, where g is the input number.

E: Alpha Sudoku

We need functions to `checkRow(int r)`, `checkColumn(int c)`, and `checkCube(int r, int c)` where `r` and `c` are row and column numbers of the top-left cell of the grid; `r,c` $\in \{0, 3, 6\}$. and we need a function `contains(String s, char c)`.

The rest is pretty straightforward.

F: Racing Around the Alphabet

First compute
`timeBetweenChars = 60*Math.PI/27.0/15`
and
`pickUpTime = 2.0`

Then, something like:

```
for(int j=0;j<len-1;j++) {
    if(line.charAt(j)==' ')
        posn1 = 0;
    else
        posn1 = line.charAt(j) - 'A' + 1;
    if(line.charAt(j+1)==' ')
        posn2 = 0;
    else
        posn2 = line.charAt(j+1) - 'A' + 1;
    int distance = posn1 - posn2;
    if(distance<0)
        distance = -distance;
```

The rest is straightforward.

G: Pals

First write the function `boolean isPal(String S)`.
The problem asks you to verify that an input string is a palindrome, or that the addition of one character will make it a palindrome. But, instead of trying to add a character you can try to remove a single character. If the string isn't a palindrome then do the following:

```
int len = s.len();
for(int j=0;j<len; j++) {
    String t = s.substring(0,j) + s.substring(j+1); // remove one char
    if(isPal(t)) {
        System.out.println("Problem " + (d++) + ": Close enough to a palindrome");
}
```

H: Slide 'Em

first decide on a structure to represent the state of the game. You could declare a class:

```
    private class State {
        int[] board;
        int move;
    }
```

You'll need four functions to manipulate the state by swapping the empty space with its north, south, east, and west neighbors.

Start with the goal state, add it to a queue and to a HashMap. Here is the HashMap declaration:

```
HashMap<State, Integer> stateTree = new HashMap<State, Integer>();
```

The integer is the number of moves to get from the goal state to that state (or vice versa).

Use Breadth-first-search to move from the goal state to all of its $9! = 362{,}880$ states.

The HashTable contains a mapping between all possible states and the minimum number of moves from each of them to the goal state.

For each dataset encode it as a State object and just look it up in the HashMap.

I: Stars

This problem is straightforward geometry:
If $i > p$, then $i = i \; mod \; p$
If $i = 0$ or $i = 1$ or $2i = p$, then there is no star.
If $2i > p$ then $i = p - i$

There will be p points on the star separated by angle:

$$\theta = \frac{2 \times \pi \times i}{p}$$

The length of each chord is then

$$l = 2 \times r \times sin(\frac{\theta}{2})$$

And the total length of all the chords is

$$p \times l$$

J: More Stars

We need to find the number of integer values of i such that $2i < p$ and i and p are co-prime. Thus, for $p = 13$, $i = 2, 3, 4, 5, 6$ all produce valid stars. For $p = 16$, $i = 3, 5, 7$ all produce valid stars.

Given p we test $i = 2, 3, \ldots p/2$. Each one that is co-prime with p produces a valid star. To test for co-primeness divide p and i by all primes in the range [2, min(i,p)]. If none of the primes evenly divide both i and p then i and p are co-prime. Since the largest value of p is 100, we need a list of primes in [2,50]. Don't generate these, just hard-code them.