Stanford ProCo

May 14, 2011

proco.
programming contest. provably cool.

Problem Packet
Novice Division

Sponsored by:

STANFORD COMPUTER SCIENCE

Palantir

Novice 2.1 **Inception** (page 1 of 1)

Overview: Print a string exactly as formatted.

Description: Once upon a time, there was a man, a man filled with dreams and hopes of a better future. But decades of living life on the edge as the best dream extractor in the world have shattered his dreams, and in fact, it has been years since this man was able to experience a dream naturally in his sleep. And even when he is able to dream with the assistance of his PASIV machine, all he sees are projections, mere shadows of what his children and late wife once were.

This man is, of course, Dominic Cobb.

Saito, rising business tycoon, is well aware of much of this. He knows that he must take down his chief business rival if he is to achieve world domination, and he cannot do so without Cobb's assistance. In order to obtain Cobb's cooperation, Saito intends to remind Cobb of how lonely he is without his children and to entice him with the opportunity to see them.

Help Saito convince Cobb to help by writing a program that will print out the following message, including quotation marks, to Cobb:

```
"DO YOU WANT TO BECOME AN OLD MAN, FILLED WITH REGRET,
WAITING TO DIE ALONE?"
```

Filename: nov21.{java, c, cpp, cc, py}

Input: There is no input for this program.

Output: The output is to be formatted exactly like the sample output given below. There are no line breaks within the output, and exactly one newline at the end of the output.

Assumptions: All input is valid.

Sample Input #1: There is no input for this program.

Sample Output #1:
```
"DO YOU WANT TO BECOME AN OLD MAN, FILLED WITH REGRET,
WAITING TO DIE ALONE?"
```

Novice 2.2 **Step up, fall down** (page 1 of 1)

Overview: See if a series of step lengths gets back to start.

Description: Cobb and his team are in a world filled with looping staircases (a set of stairs, each up or down, that loops back to the original place) but no convenient flat surface on which Cobb can spin his totem to determine whether he is in a dream or not. But as any good extractor should know, it is crucial to know if your world is reality.

In the real world, a looping staircase will go up by the same height as it goes down over the entire loop. For instance, a staircase might lead you 10 feet up, and then 10 feet down. After finishing a loop, you will end up at the same level. Such a staircase is called a *normal staircase*.

In a dream world however, normal logic does not apply! After going a loop around a staircase, you could end up at a different level from where you began. Such a staircase is called a *paradox staircase*.

You need to help Cobb determine if a staircase is a paradox before he risks his life in the world. Write a program that takes the sizes of the steps in a looping staircase and determines if it is *normal* or a *paradox*.

Filename: nov22.{java, c, cpp, cc, py}

Input: The first line contains one integer n, representing the number of steps in the staircase. The next n lines contain one integer each, representing the height of that step, and whether the step goes up or down. Positive integers represent steps up, and negative integers represent steps down.

Output: The output should consist of a single line. If the staircase is a normal staircase, print `NORMAL`. If the staircase is a paradox staircase, print `PARADOX`.

Assumptions: $1 \le n \le 1000$
The size of each step is between -1000 and 1000, inclusive.
All input is valid.

Sample Input #1:
```
3
1
3
-2
```

Sample Output #1:
```
PARADOX
```

Sample Input #2:
```
7
5
-7
1
-6
3
-1
5
```

Sample Output #2:
```
NORMAL
```

Novice 2.3 **Check your check**

Overview: Add commas to a number.

Description: In order to advance his nefarious schemes, Saito must buy the airline in which Robert Fischer Jr. will be flying, so that the Inception crew can do their job. Saito knows exactly how much money he needs to purchase the airline. He would like to do so by writing a check, but in order to write a proper check, he must include commas in the number, where required, for readability reasons. If a number has more than 3 digits, a comma needs to be placed every three digits starting from the right. For instance, 1234567 should be 1,234,567. He has hired you to automate this task for him. Good luck!

Filename: nov23.{java, c, cpp, cc, py}

Input: The first line contains a string of numbers.

Output: The first line contains the same string of numbers correctly inserted.

Assumptions: The input contains only digits.
There will be no leading 0's in the number.
The number of digits is between 1 and 1000, inclusive.
All input will be valid.

Sample Input #1: `14285714285`

Sample Output #1: `14,285,714,285`

Sample Input #2: `528`

Sample Output #2: `528`

Novice 2.4 **Delusions and dilutions** (page 1 of 1)

Overview: Calculate the duration of an event in a timeline given the same event's duration in another timeline.

Description: Yusuf (The Chemist) is developing a new drug for the team to use to put themselves and their targets into the dream world. To his annoyance, the team wants to use the drug to enter multiple levels of dreams. They intend to use the drug in each dream level to enter a deeper dream level. Doing this is complicated because time passes more slowly in with each deeper level.

The dream levels are numbered 1 and 2. Yusuf knows that *a* minutes in the real world corresponds to *b* minutes in dream level 1, and *c* minutes in the real world correspond to *d* minutes in the dream level 2. He needs to be able to convert durations from dream level 1 to dream level 2.

Filename: nov24.{java, c, cpp, cc, py}

Input: The first line will contain the numbers *a* and *b*, separated by a single space. The second line will contain the numbers *c* and *d*, separated by a single space. The third line will contain a single integer, *n*, representing the number of minutes that pass in dream level 1.

Output: The output will consist of a single number *m*, such that *m* minutes in dream level 2 correspond to *n* minutes in dream level 1.

Assumptions: $1 \le a, b, c, d, m, n \le 1{,}000{,}000$
b is a multiple of *a*, and *d* is a multiple of *c*.
m will always be an integer.
All input is valid.

Sample Input #1:
```
50 4600
35 2660
108376
```

Sample Output #1:
```
131192
```

Sample Input #2:
```
1 10
2 40
10
```

Sample Output #2:
```
20
```

Novice 5.1 **Topic nine: anagrams**

Overview: Given two strings, check whether they are anagrams of each other.

Description: One way of planting an idea into someone else's mind is to leave indirect hints of what you want the person to think. For instance, if you show the word "INCEPTION" to your debate opponents enough times, maybe they'll start thinking, "NICE POINT," which is an anagram of "INCEPTION." If you see the names of the characters Arthur, Dom, Robert, Eames, Ariadne, and Mal enough times, maybe you'll start thinking it's all "A DREAM," which is an anagram of the first initials of their names.

Help Cobb and his team complete their inception mission by checking whether they are planting the correct anagrams in their target's mind. Given a series of two strings, check whether they are anagrams of each other.

Filename: nov51.{java, c, cpp, cc, py}

Input: The input will consist of two lines. Each line will be a string. You must check whether these two strings are anagrams of each other.

Output: Output either `YES` if they are anagrams or `NO` if they are not.

Assumptions: All strings contain only uppercase letters and spaces.
Each string will have length between 0 and 20000, inclusive.
All input is valid.

Sample Input #1:
```
INTERNET ANAGRAM SERVER
I REARRANGEMENT SERVANT
```

Sample Output #1:
```
YES
```

Sample Input #2
```
ABCDEFGHIJKLMNOPQR
ZZZZZZZZZZZZZZZZZZ
```

Sample Output #2
```
NO
```

Novice 5.2 **diff dreams reality**

Overview: Take repeated differences of a sequence to find the value at which it first becomes a constant sequence.

Description: Dom and company are multiple levels down in a dream, but they want to jump back to reality. Each dream is represented by a finite sequence of numbers. They know they are in a reality if the sequence of numbers is a constant. To jump back up a level, Dom and company need to recreate the sequence of numbers representing that level.

Given a finite sequence of numbers representing a level, take the difference of consecutive integers to create a new sequence representing one level closer to reality. More precisely, the sequence representing the level above the level (x_1, x_2, …, x_n) will be (x_2 - x_1, x_3 - x_2, …, x_n - x_{n-1}). For example, jumping up one level from “5 2 8 4 9 1” will get to "-3 5 -4 5 -8". Repeating this process a finite number of times (perhaps zero) will always eventually get to a constant sequence in which all the values of the sequence are equal (perhaps of length 1).

For example, starting with the sequence “6 3 4 9”, the next sequence is “-3 1 5”, followed by “4 4”. This is the first constant sequence, and thus you should output the number 4.

Help Dom and his team find reality by computing the value constant in the first constant sequence reached by successively taking differences. This constant represents their reality.

Filename: nov52.{java, c, cpp, cc, py}

Input: The first line contains a single integer n, the length of the input sequence.
The second line contains n integers, separated by a single space, which the values of the input sequence $x_1, x_2, \ldots, x_n$.

Output: Output a single line containing a single integer, the value of the first constant sequence.

Assumptions: $1 \le n \le 10{,}000$
There will be no integer overflow in taking consecutive differences in any sequences representing any of the dream levels.
All input is valid.

Sample Input #1:
```
4
6 3 4 9
```

Sample Output #1:
```
4
```

Sample Input #2:
```
6
5 2 8 4 9 1
```

Sample Output #2:
```
-79
```

Novice 5.3 **Merging mergesort** (page 1 of 1)

Overview: Given two sorted arrays, merge them.

Description: Dom and Ariadne are in disagreement over the details of a particular dreamscape. As a compromise, they decide to each create only a part of the details of the dreamscape and then merge the two together into one. Each of them come up with a list of integers representing their respective ideas of the dream world. Help them merge them into one list representing the cohesive dreamscape.

You are given two sorted arrays of integers. Write a program that will merge both arrays into a single, sorted, output array. The merge algorithm is as follows: compare the first element of both arrays and determine the smaller element (ties may be broken arbitrarily). Place this element in the next open spot in the output array. If this element came from the first array, then the next comparison should compare the second element of the first array with the first element of the second array (and vice versa if the element came from the second array). Continue in this manner until one of the arrays runs out of numbers, in which case simply place all the numbers remaining in the other array at the end of the output array.

Filename: nov53.{java, c, cpp, cc, py}

Input: The first line contains two integers, *a* and *b*, separated by a single space, representing the number of elements in the first and second arrays. The next line contains *a* integers separated by spaces, representing the *a* elements of the first array. The next line contains *b* integers separated by spaces, representing the *b* elements of the second array.

Output: The output should a sorted list of *a*+*b* integers, printed on a single line with a single space between each integer.

The output is to be formatted exactly like the sample output given below.

Assumptions: $1 \le a, b \le 1000$
All input is valid.

Sample Input #1:
```
2 3
2 5
1 3 4
```

Sample Output #1:
```
1 2 3 4 5
```

Sample Input #2:
```
5 3
2 7 7 11 17
5 7 17
```

Sample Output #2:
```
2 5 7 7 7 11 17 17
```

Novice 5.4 **Shifting dreams** (page 1 of 1)

Overview: Rotate a permutation by one.

Description: Cobb has obtained a password to access a Swiss Bank account from his last extraction target. However, in a moment of weakness, he manages to accidentally scramble some of the digits that make up the password. He needs your help to unscramble the digits he was given in order to gain access to the account.

Cobb will give you a permutation p of the integers 0 through n - 1, inclusive. Let $p(i)$ be the i-th digit in p. So for p = 30214, then $p(0) = 3$, $p(1) = 0$, etc. Your goal is to rotate the elements in the permutation so that each index i is replaced by the element at $p(i)$. For example, after rotation, the number at index 0 will be 1, the element at index 1 will be 3, and so on.

Filename: nov54.{java, c, cpp, cc, py}

Input: The first line of input will be an integer n, representing the length of the permutation. The second line of the string will be the permutation p, where each character will be an integer between 0 and n - 1, inclusive.

Output: The output should contain the permutation after the rotation operation has been applied.

Assumptions: $1 \le n \le 10$
All input is valid.

Sample Input #1:
```
5
30214
```

Sample Output #1:
```
13204
```

Sample Input #2:
```
4
1230
```

Sample Output #2:
```
2301
```

Novice 9.1 **Black and white and gray** (page 1 of 1)

Overview: Convert from Gray code to binary.

Description: Yusuf, left behind in the first level dream, counts time using an old electromechanical clock that increments once per second. However, the clock is very unstable and will produce spurious readings sometimes, when the individual bit switches on the clock get out of synchronization. Yusuf modifies the clock to display the time in terms of the Gray code, but now he has trouble reading the numbers on the digit. Help him read the clock by converting Gray Code integers to binary.

In normal binary representation, an increment from 3 (011) to 4 (100) would involve the flipping of all 3 bit switches at once. Due to the unreliable nature of old electromechanical switches, these 3 bit switches might not change at exactly the same moment, leading to spurious results when the value is read in a transition state instead of a final state. Gray codes encode binary values such that only one bit is changed whenever a binary value is incremented or decremented.

To construct an *n*-bit Gray code, we first would need to generate a list of *(n-1)* bit Gray codes. We then create a new list of twice the length which consists of the original Gray Code list followed by the original list reversed. To generate the *n*-bit code, we prefix elements in the original list with zeros and the reversed list with ones.

For example, the 1-bit Gray code is merely the list {0, 1}, corresponding to {0, 1} in normal binary. The 2-bit Gray code can be generated by taking {0, 1}, reversing the list to be {1, 0}, concatenating the original with the reversed {0, 1, 1, 0} and prefixing the original with zeros and the new list with ones {00, 01, 11, 10}. This corresponds to the sequence {00, 01, 10, 11} in binary.

Filename: nov91.{java, c, cpp, cc, py}

Input: The first line contains the integer *n*. The second line will contain *n* bits (`0` or `1`) in an *n*-bit Gray code to be translated to normal binary.

Output: The output should contain the *n*-bit (`0` or `1`) binary representation of the input.

Assumptions: $1 \le n \le 20$
All input is valid.

Sample Input #1:
```
3
100
```

Sample Output #1:
```
111
```

Sample Input #2:
```
6
010101
```

Sample Output #2:
```
011001
```

Novice 9.2 **Flasks in flux** (page 1 of 1)

Overview: Given flasks of certain sizes, determine if you can measure a certain volume in one of the flasks.

Description: Oh no! Ariadne has taken most of Yusuf's flask set from him (something about building a dream model with glass, she says). Now Yusuf only has 2 flasks to work with, of volume *a* and *b*. Fortunately, Ariadne hasn't made off with any of Yusuf's chemical supply, so he still has an unlimited supply of his top secret dream-inducing chemical mix.

Yusuf would like to measure a certain volume of liquid, *t*, into one of the flasks, using the two flasks that he has left. During each step, he may perform any of these actions: fill up any flask to the brim with liquid, pour out all the liquid in a flask, or pour liquid from one flask to another until one of the flasks is either completely empty or completely full. You cannot pour liquid into a full flask.

For instance, given two flasks of volume 5 and 3, Yusuf can measure out 4 units of liquid by the following method.

- Fill the volume 5 flask with liquid. (5, 0)
- Pour liquid from the volume 5 flask into the volume 3 flask. (2, 3)
- Empty the volume 3 flask. (2, 0)
- Pour liquid from the volume 5 flask into the volume 3 flask. (0, 2)
- Fill the volume 5 flask. (5, 2)
- Pour liquid from the volume 5 flask into the volume 3 flask. (4, 3)

Given the desired volume and the volumes of the flasks Yusuf has left, help him determine if it is possible to obtain the desired volume in one of the flasks.

Filename: nov92.{java, c, cpp, cc, py}

Input: The input will consist of three lines with integers *t*, *a* and *b*, with one integer on each line. *t* represents the desired volume, while *a* and *b* represent the volumes of the first and second flasks, respectively.

Output: The output should consist of a single line containing `YES` if it is possible to obtain the desired volume in one of the flasks, and `NO` otherwise.

Assumptions: $1 \le a, b \le 100$
$0 \le t \le \max(a, b)$
All input is valid.

Sample Input #1:
```
4
5
3
```

Sample Output #1: `YES`

Sample Input #2:
```
5
6
4
```

Sample Output #2: `NO`

Novice 9.3 **Roundabout passage** (page 1 of 1)

Overview: Find the shortest path in a toroidal maze.

Description: Before letting Ariadne join his team as an architect, Cobb tests her by challenging her to construct a maze that he cannot solve in one minute. Ariadne cleverly designs a maze filled with twisty passages and dead ends and gives it to Cobb. Cobb is confused. To make things worse, normal rules of physics do not apply and the maze bends space. Upon reaching one side of the maze, Cobb also discovers that the building wraps around to the opposite side! Can you help him find his way out?

The maze will be represented by a rectangular grid of cells, where each cell can either be the starting position, the exit, an empty space, or an impassable wall. Each step, you can move a single space horizontally or vertically into an adjacent empty space. The edges of the maze wrap around to the opposite edges. In other words, if you are on a empty space on an edge, and the corresponding space on the opposite edge is empty, you may move to that space. Write a program to determine the minimum number of steps you need to reach the exit.

Filename: nov93.{java, c, cpp, cc, py}

Input: The first line will consist of two integers, *r* and *c*, where *r* represents the number of rows and *c* represents the number of columns in the maze.
The following *r* lines will each contain *c* characters, each representing a single cell of the maze.

- `.` - (period) An empty space
- `*` - (asterisk) Impassable wall
- `S` - (uppercase “s”) Your starting position
- `E` - (uppercase “e”) The exit

Output: The output will consist of a single integer, representing the length of the shortest path to the exit.

Assumptions: $3 \le r, c \le 50$
The maze has at least one path from the starting position to the exit.
All input is valid.

Sample Input #1:
```
3 5
*****
.S*E.
*****
```

Sample Output #1:
```
3
```

Sample Input #2:
```
7 7
*****.*
*S....*
****.**
*....**
***.*E*
*.....*
*****.*
```

Sample Output #2:
```
8
```

Novice 9.4 **Mastermind** (page 1 of 2)

Overview: Play a game of Mastermind against the machine interactively.

Description: Before anyone can successfully enter another person's subconscious mind, tough mental training must be undertaken. Ariadne, new to the Inception team, is undergoing the training, supervised by Cobb. Part of the training involves repeatedly playing the game of Mastermind one thousand times in her dream world. However, Ariadne is fed up with Mastermind by the 10th game, so she enlists you to construct a computer program to solve the next 990 Mastermind games for her.

You will play the game of Mastermind against Cobb. Cobb will select four pegs, each of which will be one of 6 possible colors (note that Cobb may select multiple pegs of the same color). You must guess the correct color and arrangement of those pegs he selected with at most twelve guesses. This is an interactive problem, where your program will iteratively make guesses, and our program (playing the role of Cobb, the code maker) will respond to each guess.

The possible colors in the game are Red, Green, Blue, Cyan, Magenta, Yellow and are represented in game as R, G, B, C, M, Y respectively. After you make each guess, the judge will inform you of *p,* the number of pegs that are both the correct color and in the correct position, and *c*, the number of pegs that of the correct color but in the wrong position.

Note that if a peg in the guess is already matched to a peg in the correct solution, those pegs cannot be matched again in counting *c*. For example, if the correct solution is RRGG and the guess is RGRR, the first position is the only one correct, so *p* is 1, and *c* is 2 because G is correct but in the wrong position, and one of the Rs is correct but in the wrong position. Note that the third R in the guess does not count towards *c*, because there are only two Rs in the correct solution. The second G in the correct solution also doesn't count towards *c*, because there is only one G in the guess. In other words, *p*+*c* is the maximum number of positions in which the correct solution and some permutation of the guess match. See the sample games for more examples.

File name: nov94.{java, c, cpp, cc, py}

Input/Output: This is an interactive problem. This means that your program will receive input from the grading environment based on the output your program produces. All input and output will be done through the console.

Output: Rules of interaction:

1. Your program should output a 4-character string consisting of only the characters (`R`, `G`, `B`, `C`, `M`, `Y`).
2. You MUST output a new line character and flush the output stream after each output! (See the sample contest problem)

3. Each query will result in two integers, *p* and *c*, with *p* representing the number of correct positions in the most recent guess, and *c* representing the number of other pegs that are the correct color but are not in the correct position.
4. Your program must terminate upon receiving a response of `4 0`, which represents a correct solution.
5. You must solve the game in at most 12 guesses and you must use at most 1 second of CPU time to solve each game. Any violation of this will result in your program being judged incorrect.

Assumptions: $0 \le p, c$
$p+c \le 4$

Sample Game #1:

```
[contestant] YYCC
[judge]      0 1
[contestant] MRRY
[judge]      1 1
[contestant] GMGY
[judge]      1 1
[contestant] RRMY
[judge]      0 2
[contestant] CRGM
[judge]      1 3
[contestant] GCRM
[judge]      4 0
```

Sample Game #2:

```
[contestant] RRRR
[judge]      2 0
[contestant] RYRY
[judge]      1 1
[contestant] GGYB
[judge]      1 1
[contestant] GGGB
[judge]      2 0
[contestant] RRGB
[judge]      4 0
```