First Bicentennial aRg Coding Competition

Submit all solutions to argcup@gmail.com
Send all questions that way as well

Buzz Kill

PROBLEM:
You've probably heard of the famous "Fizz Buzz" problem, which, supposedly, 99.5% of programmers can't solve in an interview. This is a little crazy, so let's run a closed experiment.

Solve the Fizz Buzz problem!

'Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz"'

INPUT: N/A

OUTPUT:
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
...

Collatz

PROBLEM:
As you may be able to tell, one of the organizers of this competition is a math major. Also, that guy did research on the Collatz, so here's another mathy question!
The Collatz conjecture asks the following question: "Say you have a number **n**, if **n** is even, return **n/2**. If **n** is odd, return **(3 * n) + 1**. If I keep repeating this process, eventually I'll get back 1 for all positive integers."

For example, say we start with 3. The steps of the Collatz are as follows:
3 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1
Meaning we repeat the Collatz function 7 times before we hit 1.

For a given positive integer **n**, return the number of steps until **n** is 1.

INPUT: 3 (passed in as an argument)

OUTPUT: 7

Twin Primes

PROBLEM:
The Twin Primes conjecture in mathematics asks if there are an infinite number of primes p such that p+2 is also prime. For example, 3 and 5 are twin primes, as are 5 and 7.

This is a pretty hard question, so we'll scale it back. Can you print the 100th pair of Twin Primes? Remember. Your code has to run in under 8 seconds, so make sure that you're comfortably under that!

Example: Printing the 3rd pair of Twin Primes would yield
11, 13

INPUT: (NONE)

OUTPUT: p, p+2

Closest Pairs

PROBLEM:
Sometimes couples are easy to pick out. Sometimes, they aren't.

Given a list of integers, return the two numbers that have the smallest absolute difference.

INPUT: (Passed in as an argument)
-20 -3916237 -357920 -3620601 7374819 -7330761 30 6246457 -6461594 266854

OUTPUT:
-20, 30

Tortoise and the Hare

PROBLEM:
Despite what certain morality stories may have you believe, hares are a lot faster than tortoises. Fast enough in fact that hares really are not the best kind of people in races.

In every race between a tortoise and a hare, the hare always wins. *Always*. So much so, that in certain former Eastern Bloc racing circuits, people only keep track of the checkpoints the hare passes.

Remember when I said that hares aren't the best kind of people? They're not even the best kind of hares. In some of these races, rather than simply go to the end and stop, the hares will circle back around at some point and taunt their shelled opponent by circling. Forever. And they still win.

Given **n** races, where each race has **m** checkpoints. For each of those **m** checkpoints, we have a pair of space separated checkpoint names, indicating where the hare moved from and to. Try to determine whether or not the hare circles back around. If they don't, print **NOT A BAD GUY**

If they do circle around at some point, print **WHY MUST YOU TAUNT ME SO AT _**, where the _ is the name of the checkpoint that is at the beginning of the hare's vicious cycle, as seen from the start point. (That is to say, it's the first checkpoint in the cycle we come across as the noble tortoise.)

(Sample inputs on next page)

(Possible Problem: Medium) Tortoise and the Hare

INPUT: (file: **tortoisehare.txt**)

```
4
7
A B
B C
C D
D B
B C
C D
D B
5
A B
B C
C D
D E
E F
1
A B
6
1 2
2 3
3 4
4 2
2 3
3 4
```

OUTPUT:

```
WHY MUST YOU TAUNT ME SO AT B
NOT A BAD GUY
NOT A BAD GUY
WHY MUST YOU TAUNT ME SO AT 2
```

Consonant Convictions

PROBLEM:

In English, there are 26 letters that are either **vowels** or **consonants**. In this problem, we consider **a, e, i, o,** and **u** to be vowels, and the other 21 letters to be consonants.
A tribe living in the Greatest Colorful Jungle has a tradition of naming their members using English letters. But it is not easy to come up with a good name for a new member because it reflects the member's social status within the tribe. It is believed that the less common the name he or she is given, the more socially privileged he or she is.

The leader of the tribe is a professional linguist. He notices that hard-to-pronounce names are uncommon, and the reason is that they have too many **consecutive consonants**. Therefore, he announces that the social status of a member in the tribe is determined by its **n**-value, which is the number of substrings with at least **n** consecutive consonants in the name. For example, when **n** = 3, the name "quartz" has the **n**-value of 4 because the substrings **quartz**, **uartz**, **artz**, and **rtz** have at least 3 consecutive consonants each. A greater **n**-value means a greater social status in the tribe. Two substrings are considered different if they begin or end at a different point (even if they consist of the same letters), for instance "tsetse" contains 11 substrings with two consecutive consonants, even though some of them (like "**tse**tse" and "tse**tse**") contain the same letters.

All members in the tribe must have their names and **n** given by the leader. Although the leader is a linguist and able to ensure that the given names are meaningful, he is not good at calculating the **n**-values. Please help the leader determine the **n**-value of each name. Note that different names may have different values of **n** associated with them.

$1 \leq \mathbf{T} \leq 100$.
$0 < \mathbf{n} \leq \mathbf{L}$.

Consonant Convictions

INPUT: (consonants.txt)
The first line of the input gives the number of test cases, **T**. **T** test cases follow. The first line of each test case gives the name of a member as a string of length **L**, and an integer **n**. Each name consists of one or more lower-case English letters.

3
quartz 3
gcj 2
tsetse 2

OUTPUT:
For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is the **n**-value of the member's name.

Case #1: 4
Case #2: 3
Case #3: 11

Max A

PROBLEM:

Imagine you have a special keyboard with following four keys:

- Key 1: Prints 'A' on screen
- Key 2: (Ctrl-A): Select screen
- Key 3: (Ctrl-C): Copy selection to buffer
- Key 4: (Ctrl-V): Print buffer on screen appending it after what has already been printed.

If you can only press the keyboard for N times (with the above four keys), write a program to produce maximum numbers of A's.

INPUT:

Each line will contain the number of keystrokes you are allotted, and output should correspond to the maximum number of A’s you can produce.

3
7
10

OUTPUT:

3
9
20

Almost Sorted

PROBLEM:

Given an integer array of size N, is it possible to sort the array in ascending order using only one of the following operations once:

-Swap a single pair of elements.

-Reverse a single subsegment of the array.

INPUT: (file: **almost.txt**)

Each line of the input will contain an integer array separated by spaces, where the first line is the number of lines to follow.

3
4 2
3 1 2
1 5 4 3 2 6

OUTPUT:

The output should be either S, R, or N, depending on if the array can be sorted by a swap, by a reverse, or is not almost sorted.

S
N
R

Number Puzzles

PROBLEM:
Dear god. Will these numbers ever end?

Sometimes when you're stuck on a problem set, it's best to just assume the answer, and then derive your way blindly from both sides until they meet awkwardly in the middle.

Specifically, if you're in MATH 777 "The Mystical Properties of 7", your main goal is to find some combination of 7's and simple arithmetic operators that get you to some value. You're allowed parenthesization, division, addition, subtraction, and multiplication, as well as any number of 7's.

The first integer will be **n**, the number of values that follow. Given **n** values
Print "YES: EQN" if there's some expression involving 7's that equal that value, and replace EQN with the expression.
Otherwise, print "NO 7 DOUBLE WHAMMY".

INPUT: (file: **sevens.txt**)
3
7
42
31

OUTPUT:
YES: 7
YES: (7 * (7 + (7 - 7)) - (7 * (7 / 7)))
NO 7 DOUBLE WHAMMY

Longest Common Substring

PROBLEM:

You are given two strings, you want to find the length of the longest common substring. For example, consider the strings “biggest” and “jester”, the LCS would be “est” and your output should be 3.

INPUT: (file: **longestcommon.txt**)

Each line will contain a pair of strings separated by a space, where the first line is the number of lines to follow.

3
cow caterpillar
foolish barstool
politician polite

OUTPUT:

1
3
5

Gold Efficiency

PROBLEM:

Occasionally in video game worlds, players are offered the chance to trade their gold for items of great power. Say you have just come across such a shop, and have your choice of wands, swords, or armor. Wands give you Magic (MGK) stat, swords give you Attack (ATK) stat, and armor gives you Defense (DEF) stat. For each type of item there are 4 tiers, common, rare, epic and legendary which give bonuses of +1, +2, +3, and +4 and also cost 100, 175, 225, and 250 gold respectively.

You can buy as many items as you want, and there is an endless supply of each type and tier. Given some amount of gold, and certain starting stats, what items do you buy to maximize your strength? Strength is given by: STR = log(MGK) + log(ATK) + log(DEF).

INPUT:

The first line of input will contain the number of cases. Each case will have four lines. The first line will contain your gold, while the second through fourth will contain a stat name, (MGK, ATK, DEF) followed by a space, and then your current stat:

1
255
MGK 5
ATK 18
DEF 2

OUTPUT:

For each input case, there should be a single line of output, containing the names of the item bought separated by commas. Items should be described by rarity, then a space, then their type:

legendary armor

Subset Components

PROBLEM:

You are given an array with n 64-bit integers: d[0], d[1], ..., d[n - 1].

BIT(x, i) = (x >> i) & 1. (where B(x,i) is the ith lower bit of x in binary form.)

If we regard every bit as a vertex of a graph G, there exists one undirected edge between vertex *i* and vertex *j* if there exists at least one k such that BIT(d[k], i) == 1 && BIT(d[k], j) == 1.

For every subset of the input array, how many connected-components are there in that graph?

The number of connected-components in a graph are the sets of nodes, which are accessible to each other, but not to/from the nodes in any other set.

For example if a graph has six nodes, labelled {1,2,3,4,5,6}. And contains the edges (1,2), (2,4) and (3,5). There are three connected-components: {1,2,4}, {3,5} and {6}. Because {1,2,4} can be accessed from each other through one or more edges, {3,5} can access each other and {6} is isolated from everyone else.

You only need to output the sum of the number of connected-component(S) in every graph.

INPUT:

Input will consist of one line with the number of integers in the array d. The next line will contain the array elements separated by a space.

3
2 5 9

OUTPUT:

504

Duplicate

PROBLEM:

You are given an array of integers of size N, where each integer is positive and less than some constant N. Find an occurrence of a duplicate integer in the array. You should use no more than O(1) space.

INPUT:

The input will consist K cases, where each case has one line containing the array of integers, separated by a space:

1 2 4 2 5
3 1 7 2 6 4 4 1
6 3 7 1 2 4 5 5

OUTPUT:

2
1 or 4
5

Amnesiac Maze

PROBLEM:

Some games involve dungeons. In a lot of these dungeons, you're popped off in the middle of somewhere and you want to get to a specific location in order to progress. Sometimes, if the developer is particularly sadistic, you'll have no minimap, and each room will be indistinguishable.

As an intelligent, rational person, when I come upon these occasions I do the rational, intelligent thing. I do an exhaustive search of the space in the hopes of blindly stumbling upon the exit. However, my memory isn't that good. Especially when every room looks the same. When you're given a maze of some dimension D, can you write a program to check if a target Y is reachable from your initial position P? Your solution must use O(1) memory, not including reading in the maze. You are not allowed to modify values in the maze, as this would be considered an O(D^2) memory solution.

Additionally, you're only able to move in the cardinal directions: up, down, left, or right.

Your program will be passed in:
N, the number of mazes to follow, 1 <= N <= 10
D, the dimensions of the maze, 1 <= D <= 8
D lines of D characters, where an O is an open space, and X is a wall, P is the player, and Y is the target location.

SAMPLE INPUT: (file: **maze.txt**)

```
2
3
OOO
OPX
OOY
4
XXXX
XOXY
XOXO
XPOX
```

SAMPLE OUTPUT:

```
YES
NO
```