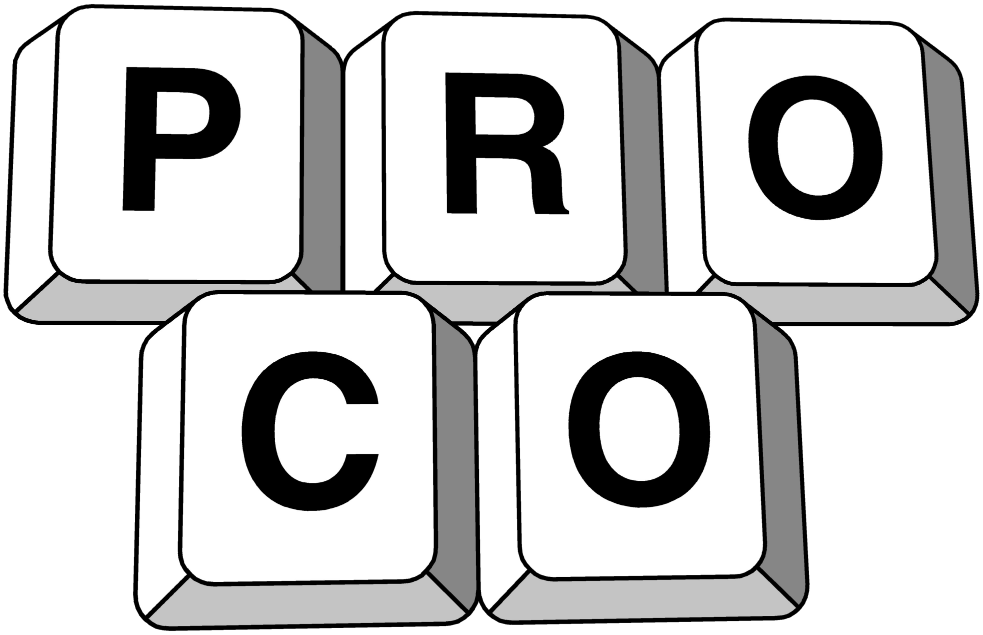
Stanford
High School
Programming Contest



May 16, 2009

Problem Packet

Problem 2.1 **Jinja Ninjas vs. Irate Pirates** (page 1 of 1)

Overview: Output exactly what is given in the sample output below.

Description: Ninjabay, the world's largest uberbytetorrent tracker based in Jinja, Uganda, recently suffered a devastating loss at the hands of Tripoli's judicial court system. Throw in the current credit crunch and other factors beyond its control, and Uberbytetorrent is now poised to lose a whopping fifty percent of its revenue this quarter. And all because some random employees from Ninjabay wanted to pick a fight with a band of innocent-looking pirates…

Needless to say, Uberbytetorrent is not pleased. Many a quasi-legal company might simply cut their losses and move on, but Uberbytetorrent has never been, and never will be, Just Another Company.

Demonstrating a dubious amount of professionalism, the president of Uberbytetorrent decided to exact revenge on the hapless Ninjabay employees who caused this entire mess, but unfortunately experienced a burst of guilt right after phoning his BFF, Chicago's most notorious mafia boss.

As such, he would like to give the employees a chance to leave Ninjabay and work for him instead. The mafia goons have been instructed to transmit a particular message to all of the employees, but alas, the vast majority of them are high school dropouts who have never heard of computer science. And this is where you come in: write a program to output the given message exactly, and your life just might be spared. Stay safe!

Time Allocation: 1 second

Input: None

Output: The output should consist of two lines of text, formatted exactly like the sample output given below.

Assumptions: No input will be given.

Sample Output:

```
I\n a\n i\na\ne e\ra, o\bey \no o\ne o\n ou\r i\n\te\r\ne\t;
\tu\r\n \to U\be\r\by\te\to\r\re\n\t.
```

Problem 2.2 **Nitwitter** (page 1 of 1)

Overview: Check if a base ten number is a power of two.

Description: Nitwitter, the fastest growing microblogging site around, recently hired several aliens from the enchanted planet of Tarnia to market its services to the inhabitants of Tarnia. The fact that each alien only has two fingers and no mouths makes interspecies communication somewhat difficult. However, a group of industrious researchers at Nitwitter has developed a revolutionary method of passing colored tokens back and forth between the species to convey information.

Tokens are carefully stacked and packed into boxes before being shipped to the destination planet. For storage purposes, it is optimal to begin transportation for *n* tokens, where *n* is a power of two. In fact, if *n* were not a power of two, such transportation would be virtually impossible. As an example, 32 (i.e. 2^5) tokens could be easily transported, but sending 37 (i.e. $2^5 + 5$) tokens would be daunting task that not even the brave of researchers at Nitwitter dare tackle.

Nitwitter needs your help to determine whether a given number of tokens could be easily sent to Tarnia. You should read in the proposed number *n* of tokens from the console, and output `yes` if *n* is an integral power of two, and `no` otherwise. Good luck!

Time Allocation: 1 second

Input: The input consists of a single integer *n*, signifying the number of proposed tokens Nitwitter desires to ship to Tarnia.

Output: The output should consist of a single lowercase word, either `yes` or `no`, indicating whether *n* is an integral power of two.

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

Assumptions: *n* will be an integer between 1 and 65,536, inclusive.
An integral power of two is defined as any integer *m* of the form 2^x, where *x* is an integer greater than or equal to 0.
All input will be valid.

Sample Input #1:

```
1
```

Sample Output #1:

```
yes
```

Sample Input #2:

```
13
```

Sample Output #2:

```
no
```

Problem 2.3 ATnTA (page 1 of 1)

Overview: Verify if the letters in a string form a palindrome.

Description: Communications company ATnTA has been responsible for handling a lot of the traffic on the Internet. However, more and more information is being sent online, and all the data has been clogging the Tubes recently. In order to make sure that their data is getting transmitted correctly, ATnTA has decided to send their data in redundant palindromic form. That way, it is possible to check if the data arrived correctly, by going through it backwards and verifying that it's the same in reverse.

A word is a palindrome (i.e. is palindromic) if it is a word which remains unchanged when its letters are reversed. For this problem, ATnTA considers a palindrome to be any string whose letters form a palindrome. Thus, the string should become a palindromic word after

1) removing spaces, and
2) converting every letter to lowercase.

Thus, the following are palindromes:

- `A man a plan a canal Panama`
- `ProCo is I O Corp`
- `Sq a blooL baQs`
- `Hannah`
- `RaCeCAR`

Time Allocation: 1 second

Input: The input consists of a string *c* on a single line.

Output: The output should consist of a single word `yes` or `no`, indicating whether *c* becomes a palindromic word after removing all spaces and converting every letter to lowercase.

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

Assumptions: *c* will contain between 1 and 1,000,000 characters, inclusive.
c will contain only uppercase and lowercase letters and spaces.
c will begin and end with a letter.
All input will be valid.

Sample Input #1:

```
Go hang a salami I m a lasagna hog
```

Sample Output #1:

```
yes
```

Sample Input #2:

```
Reversed deserver
```

Sample Output #2:

```
no
```

Problem 2.4 Most Ostentatiously Dressed Employee (page 1 of 1)

Overview: Find the most frequently occurring element in a list.

Description: Moogle, the world's fastest search engine and provider of a host of web applications that have nothing to do with web search, is determined to reclaim its #1 position on Fortune's Best Companies to Work For list after being replaced by some upstart company that stores and manages network data. To improve the Moogler experience and generate more press, Moogle has decided to hold a company-wide beauty contest, in which each employee votes for which Moogler they find most attractive.

Unfortunately, California's first summer blackout of the year hit just as the program to tabulate results was running, permanently frying one of Moogle's indestructible servers and killing the program that was running.

You are interviewing for a software engineering position at Moogle, and as the final question, you have been asked to reconstruct the program that was used to determine the winner for the beauty contest. Do you have what it takes to become a Moogler?

Time Allocation: 1 second

Input: The first line contains two integers n m, separated by a single space, where n is the number of votes that were cast, and m is the number of employees currently at Moogle. The next n lines each contain one positive integer j, representing a vote for Moogle's j-th employee.

Output: The output should consist of a single integer w, representing the employee number of the person with the most votes.

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

Assumptions: n will be an integer between 1 and m, inclusive.
m will be an integer between 1 and 100,000, inclusive.
Each j will be an integer between 1 and m, inclusive.
There will be exactly one employee with the most votes.
All input will be valid.

Sample Input #1:

```
9 15
11
2
5
2
7
10
3
7
7
```

Sample Output #1:

```
7
```

Sample Input #2:

```
9 10
1
2
3
2
2
3
1
1
1
```

Sample Output #2:

```
1
```

Problem 2.5 Snazzy snApply snuppercase (page 1 of 1)

Overview: Capitalize every word.

Description: snApple recently held its biannual Bring Your Daughter to Work day, during which hordes of young, wide-eyed schoolgirls follow their parents around and try not to destroy everything in sight.

Unfortunately for the snIphone division, one vindictive second grader who failed her spelling and grammar quiz last Tuesday (and was fearful of her chances of getting into Stanford as a result) was on a mission: to destroy the evil of Capital Letters, for once and for all.

Truly the daughter of two revered software engineers, the girl released a virus on all snIphone department computers that turned every sentence on the snIphone Application Store site to lowercase. Eve Mobs, the CEO and founder of snApple, is appalled by the aesthetically displeasing result and is desperately in need of a solution!

Kind soul that you are, you have generously volunteered to fix this problem for her, and in an effort to compensate for the missing capital letters, you plan to capitalize the first letter of each and every word instead. Good luck!

Time Allocation: 1 second

Input: The input consists of a single sentence *s* on one line. *s* may contain any number of words. Adjacent words are separated by a single space.

Output: The output should consist of the input *s* with the first character of each word capitalized.

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

Assumptions: *s* will contain between 1 and 1000 characters, inclusive.
s will contain only spaces and the lowercase alphabet letters `a-z`.
s will begin and end with a letter.
All input will be valid.

Sample Input #1:

```
we provide great gifts for every occasion
```

Sample Output #1:

```
We Provide Great Gifts For Every Occasion
```

Sample Input #2:

```
stanford students receive a discount
```

Sample Output #2:

```
Stanford Students Receive A Discount
```

Problem 5.1 Stock-Revealing Backroom Dealing (page 1 of 1)

Overview:

Interactively guess a number based on knowledge that the desired number is less than, greater than, or equal to the guesses.

Description:

Nagilent and PAMD, both well-established companies in the technology industry, are considering the possibility of merging. The CEO of Nagilent is bartering some Nagilent stock for PAMD stock with the CEO of PAMD, in order to artificially boost stock trading for both companies. It is possible for the two CEOs to trade anywhere from 1 to *n* shares of stock, but the CEO of PAMD only wants to trade an unknown number *m* shares of stock and will only tell the CEO of Nagilent whether his offer is too high, too low, or right on the dot. The CEO of Nagilent has requested your assistance in this latest endeavor of questionable business ethics.

After we give you *n*, you are allowed to query us about any number *x* within the range of 1 to *n* and we will tell you if *m* is less than, equal to, or greater than *x*. Your task is to use this information to successfully query *m*. Good luck!

Time Allocation:

1 second

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.

Rules of interaction:

1. Your program should begin by reading in a single integer *n*.
2. Then your program should output an integer *x*, representing a query about the hidden integer *m*.
3. You MUST output a new line character and flush the output stream after each output!
4. Each query will result in an integer response *k*, which will be either `-1`, `0`, or `1`, where `-1` indicates that *m* is less than *x*, `1` indicates that *m* is greater than *x*, and `0` indicates that you have successfully queried *m*.
5. You can make as many queries as you want, but your program should terminate after *m* has been successfully queried.

Assumptions and Expectations:

n will be an integer from 1 to 1,000,000,000 inclusive.
m will be an integer from 1 to *n*, inclusive.
x must be an integer from 1 to *n*, inclusive.
If any output is invalid, your program will be deemed incorrect.

Sample Run:

```
 Input: 4
Output: 2
 Input: 1
Output: 3
 Input: 0
```

Problem 5.2 **All Yar Whole World Are Belong To YaWho?** (page 1 of 1)

Overview: Determine whether a given string is a substring of another.

Description: YaWho?, in desperation to out-compete its rival search engine Moogle, has decided to implement a new system to guess search terms from poorly spelled user input. After several studies of search terms, YaWho? has determined that the most common user typo is adding additional characters to the search phrase. They already have a program to guess possible words based on incorrectly spelled input, but they need you to write a program to verify whether or not a given word is a possible search term.

A word is a possible search term only if all of its letters appear in order within the search phrase, but the characters do not need to be adjacent to each other. For example, `abc` and `abd` are possible search terms for the search phrase `abcd`, but `cde` and `bca` are not.

Time Allocation: 1 second

Input: The first line contains the string *s*, which represents the search phrase. The second line contains the string *t*, which represents the search term under consideration.

Output: The output should consist of a single lowercase word, either `yes` or `no`, indicating whether *t* is a substring of *s*.

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

Assumptions: *s* and *t* will contain between 1 and 1000 characters, inclusive.
s will contain only lowercase letters and spaces.
t will contain only lowercase letters (no spaces).
s and *t* will each begin and end with a letter.
All input will be valid.

Sample Input #1:

```
stanford proco
foo
```

Sample Output #1:

```
yes
```

Sample Input #2:

```
stanford proco
bar
```

Sample Output #2:

```
no
```

Problem 5.3 See No Evil, Speak No Evil (page 1 of 1)

Overview: Compute the *n*-th look-say-reverse number.

Description: Great news! The jRobot Corporation, widely celebrated for its self-vacuuming Zoomba robot, is on the verge of completing its prototype LISA (Lactose Intolerant Super Automaton) robot, a friendly, self-learning machine that is sure to win even the most skeptical of technophobes over with its human-like charms.

But the research team seems to have run into a slight snag: while the rate of LISA's emotional development is off the charts, she has formed an odd affinity for look-say numbers, and the rest of her mathematical processes have shut down. LISA is well aware of this problem and is dangerously close to self-destructing in desperation. To prevent this catastrophe from occurring, you have been hired to repair LISA's glitch.

Clearly, the best course of action is to reverse LISA's fondness for look-say numbers by having her develop an attachment for "look-say-reverse" numbers as well and having the attachments cancel each other out. To introduce these numbers to her, you have been asked to generate the *n*-th look-say-reverse number. Are you up to the task?

The first look-say-reverse numbers are `1`, `11`, `12`, `2111`, `1321`, `11213111`, and `1331112112`. If we know a look-say-reverse number, we can get the next one by splitting the number into consecutive runs of identical digits (so `11213111` becomes "11 2 1 3 111"), saying how many of each digit there are (e.g., "two 1, one 2, one 1, one 3, three 1"), writing this down as a new number ("2112111331"), and reversing it (to get `1331112112`).

Time Allocation: 1 second

Input: The input consists of a single integer *n*.

Output: The output should consists of a string *s* on one line, representing the *n*-th look-say-reverse number.

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

Assumptions: *n* will be an integer between 1 and 49, inclusive.
s will contain no more than 1,000,000 digits.
All input will be valid.

Sample Input #1:

```
4
```

Sample Output #1:

```
2111
```

Sample Input #2:

```
7
```

Sample Output #2:

```
1331112112
```

Problem 5.4 Livebold's Lapse: A Powerful Predicament (page 1 of 1)

Overview: Print out all the powerful numbers within a range of integers.

Description: Voting machine company Livebold has discovered a security hole in their machines! In 2008, a few nefarious individuals discovered that certain numeric passcodes would allow access to vote counts for districts all across the country. As it turns out, these special numbers, called powerful numbers, all have a certain property.

A powerful number is a positive integer *m*, such that for every prime number *p* dividing *m*, p^2 also divides *m*. In order to fix this breach, Livebold needs to be able to find all the powerful numbers in a certain range, and they have enlisted you to help. The fate of American democracy lies in your capable hands!

Time Allocation: 1 second

Input: The first line contains an integer *a*, representing the lower bound of the range. The second line contains an integer *b*, representing the upper bound of the range.

Output: The output should be a listing of all the powerful numbers between *a* and *b*, inclusive. Each powerful number should be on a separate line.

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

Assumptions: *a* will be an integer between 1 and 100,000, inclusive.
b will be an integer between *a* and 100,000, inclusive.
All input will be valid.

Sample Input #1:

```
30
80
```

Sample Output #1:

```
32
36
49
64
72
```

Sample Input #2:

```
110
190
```

Sample Output #2:

```
121
125
128
144
169
```

Problem 5.5 Viral Spiral: A Concern at CERN (page 1 of 1)

Overview: Output the sum of all neighbors of a cell in an *n* x *n* number spiral.

Description: The aptly-named *Spiroviro* virus has infected the computer grid at CERN. Starting from the northwest corner of the basement computer cluster, *Spiroviro* proceeds to spread clockwise along the computers on the edge, which have lower priority access to the networked antivirus scan. Each new infection causes exactly one more damage than the last, as measured by a special BleederMeterReader.

Given a size *n* and a computer at location (*x*, *y*), where *x* increases to the south and *y* increases to the east, help CERN determine the sum of the damage dealt by *Spiroviro* to neighboring computers. For example, for *n* = 5, the computer grid would exhibit a damage pattern of

```
 1  2  3  4  5
16 17 18 19  6
15 24 25 20  7
14 23 22 21  8
13 12 11 10  9
```

with (1, 1) = 1, (3, 5) = 7, and (4, 2) = 23. Neighboring computers are defined in the cardinal directions. That is, the neighbors of (1, 1) are (1, 2) and (2, 1); the neighbors of (4, 2) are (3, 2), (4, 1), (4, 3), and (5, 2).

Time Allocation: 1 second

Input: The first line contains an integer *n*, representing the side length of the square computer grid. The second line contains two integers *x y*, separated by exactly one space.

Output: The output should consist of a single integer *k* representing the sum of the damages dealt to all computers neighboring the (*x*, *y*) location in the *n* x *n* computer grid.

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

Assumptions: *n* will be an integer between 1 and 1000, inclusive.
x and *y* will each be an integer between 1 and *n*, inclusive.
k will be an integer between 1 and 4,000,000, inclusive.
All input will be valid.

Sample Input #1:

```
5
4 2
```

Sample Output #1:

```
72
```

Sample Input #2:

```
5
4 1
```

Sample Output #2:

```
51
```

Problem 9.1 Pewlett Hackered Printer Problem (page 1 of 2)

Overview: Interactively find the missing number in an array by querying bits.

Description: Pewlett Hackered (PH), the famous printer company, just released the brand-new PH-x series! These printers can print a bedazzling variety of colors. The PH-x256, for example, has 256 cartridges of colors, numbered 0 through 255. In order to maintain their vibrant PH-level of printing and keep shady companies from producing imitations, the cartridges are arranged in a random order inside the printer.

You recently purchased a PH-x*n* printer containing *n* cartridges, but it seems that some PH employees fell asleep on the job, so exactly one random cartridge of every new PH-x printer is missing from each model (including your PH-x*n* printer). The super-sensitive security mechanisms prevent anyone from opening the printer after it has been manufactured, so you can't look inside to find the number of the missing cartridge.

However, it turns out that it is actually possible to find out information about the cartridges by querying the printer for a bit of information at a time. This can be done up to 2*n* times on a PH-x*n* until the patented PH-Copy-Safe™ self-destruct mechanism activates. Can you find the number of the missing cartridge and salvage your printer?

Each PH-x*n* printer is designed to contain exactly *n* cartridges in random order, where *n* is a power of 2. Inside a broken PH-x*n* printer, exactly one cartridge is missing, so there are only *n*-1 cartridges present. The cartridges are stored in a secret array with *n*-1 distinct integers from 0 through *n*-1. Your program must find out which cartridge number is missing by querying the *j*-th binary digit of the *i*-th array element. We will return the *j*-th digit indexing from the least significant digit (i.e. counting digits from the right) at 0. You can make up to 2n queries, after which you must output the number of the missing cartridge.

For example, consider the array {1, 3, 0, 7, 2, 5, 4} for *n*=8 and *m*=6. In binary, this array holds {001, 011, 000, 111, 010, 101, 100}. Here are some sample queries (output) and responses (input):

Output	Secret array (behind the scenes)	Input
`0 0`	{00**1**, 011, 000, 111, 010, 101, 100}	`1`
`4 0`	{001, 011, 000, 111, 01**0**, 101, 100}	`0`
`4 1`	{001, 011, 000, 111, 0**1**0, 101, 100}	`1`
`4 2`	{001, 011, 000, 111, **0**10, 101, 100}	`0`
`6 2`	{001, 011, 000, 111, 010, 101, **1**00}	`1`
`7 2`	{001, 011, 000, 111, 010, 101, 100}	Program terminated (Out of bounds)
`-1 0`	{001, 011, 000, 111, 010, 101, 100}	Program terminated
`3 8`	{001, 011, 000, 111, 010, 101, 100}	`0` (All numbers have infinite leading 0's)
`3 -1`	{001, 011, 000, 111, 010, 101, 100}	Program terminated

Problem 9.1 **Pewlett Hackered Printer Problem**

Time Allocation: 1 second

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.

Rules of interaction:

1. Your program should begin by reading in a single integer *n*.
2. Then, for each of up to 2*n* queries, your program should output a single pair of integers *i j*, separated by a single space.
3. You MUST output a new line character and flush the output stream after each output!
4. Each query will result in an integer response *k*, representing the *j*-th binary digit of the *i*-th element of the secret array.
5. At any time in place of the 2*n* queries, or immediately following the 2*n*-th query, your program may output a single integer *m* representing the missing cartridge. This should be the final output from your program.

Assumptions and Expectations:

n will be an integral power of 2, from 2 to 4096, inclusive.
i should be an integer between 0 and *n*-2, inclusive.
j should be an integer greater than or equal to 0.
The response *k* will be either `0` or `1`.
If any output is invalid, your program will be deemed incorrect.
Only one attempt at *m* is allowed.
No more than 2*n* queries will be accepted. After 2*n* queries, your program must output a single integer *m*. Otherwise, your program will be terminated and deemed incorrect.

Sample Run:

```
 Input: 4
Output: 0 0
 Input: 1
Output: 0 1
 Input: 1
Output: 1 0
 Input: 0
Output: 1 1
 Input: 0
Output: 2 0
 Input: 0
Output: 2 1
 Input: 1
Output: 1
```

(The cartridge array was {3, 0, 2}, or {11, 00, 10} in binary.)

Problem 9.2 **Big Green, Bankruptcy, and the Giant Maze of Doom** (page 1 of 2)

Overview: Find the length of the shortest path between two locations in a maze of numbers.

Description: Big Green corporation really, *really* doesn't want to file for Chapter 7 bankruptcy. In desperation, it has decided to kidnap employees from rival corporations and give them an offer they can't refuse: sabotage the companies they work for, or get thrown into the Giant Maze of Doom (GMD). For decades, many believed the GMD to be a mere urban legend, one of those stories mothers told their children back in the day to keep them in line. Alas, for the handful of loyal software engineers from the rival corporation, the GMD is, without a doubt, pure reality.

The GMD is divided into *r* rows and *c* columns of squares. There is an integer on each square, representing the number of squares the engineers must move in any cardinal direction (up, down, left, or right) to avoid the land mines and fire-breathing dragons. Surrounding the GMD is a bottomless pit, so moving past the edge of the maze is not recommended. If a move would put an engineer out of bounds, he or she cannot move at all in the direction.

But however bleak life is looking for these faithful employees, hope is in the air! If the engineers land on a special square, a trapdoor will be revealed and they will fall back into the real world. As one of these trustworthy engineers, you are the only one with a laptop, and thus the only one that can write a program that will determine the minimum number of moves necessary to escape the GMD and spread news of Big Green's evil plans to the world. The fate of our technological future rests in your hands – good luck!

Consider the following 3 x 5 GMD configuration:

```
12123
02314
21235
```

Locations (*x*, *y*) are numbered starting at (1, 1) in the upper-left corner, with *x* increasing down and *y* increasing to the right. Some locations and their valid moves are listed below:

Location	Value	Valid moves
(1, 1)	1	Right (1, 2); down (2, 1)
(3, 3)	2	Left (3, 1); up (1, 3); right (3, 5)
(2, 5)	4	Left (2, 1)
(2, 4)	1	Left (2, 2); up (1, 3); right (2, 4); down (3, 3)
(2, 3)	3	None
(2, 1)	0	None
(3, 5)	5	None

The minimum number of moves from (1, 1) to (3, 5) is 4: right (1, 2), down (3, 2), right (3, 3), right (3, 5).

Problem 9.2 **Big Green, Bankruptcy, and the Giant Maze of Doom** (page 2 of 2)

Time Allocation: 1 second

Input: The first line of the input contains two integers *r c*, separated by exactly one space, representing the number of rows and columns in the maze.

The next *r* lines each contain a string of *c* integers, representing the sequence of values in each location of the *r*-th row of the maze.

The next line contains two integers x_{start} y_{start}, separated by exactly one space, representing the row and column index of the starting location.

The last line contains two integers x_{end} y_{end}, separated by exactly one space, representing the row and column index of the ending location.

Output: The output should consist of a single integer *k* representing the number of moves needed to get from the starting location to the ending location.

If the starting and ending locations are the same, output `0`. If there is no possible path from the starting to the ending location, output `-1`.

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

Assumptions: *r* and *c* will each be an integer between 1 and 100, inclusive.
The sequence of digits will contain only the digits `0123456789`, with each digit representing exactly one square.
x_{start}, y_{start}, x_{end}, and y_{end} will each be between 1 and 100, inclusive. The starting location and ending location may be the same.
All input will be valid.

Sample Input #1:

```
2 2
11
01
1 1
2 2
```

Sample Output #1:

```
2
```

Sample Input #2:

```
3 4
1023
0112
3211
1 1
1 3
```

Sample Output #2:

```
-1
```

Problem 9.3 **Deranged Derangements** (page 1 of 1)

Overview: Print out the *k*-th lexicographical derangement of the first *n* letters.

Description: MyFace (*MyFace makes the world a better place!*), the fastest growing online social network, is rolling out a new version of their online social networking website interface. As per tradition, every feature must be placed in a different location on the web page than in the previous version for maximum customer confusion. There is a certain number *n* of features, up to 26, conveniently labeled A, B, C, D, etc. From these features, there are a finite number of rearrangements, specifically called derangements, that can be made.

Derangements are permutations of a sequence where each letter in the permutation is not in the same location as in the starting sequence. For example, the derangements of ABCD in lexicographical (alphabetical) order are BADC, BCDA, BDAC, CADB, CDAB, CDBA, DABC, DCAB, and DCBA. ACBD is not a derangement because the A is in the same location as the A in ABCD. Your task is to find the *k*-th lexicographical (in alphabetical order) derangement of MyFace features. The future look of MyFace depends on you, so be careful!

Time Allocation: 1 second

Input: The first line contains an integer *n*, indicating that the starting sequence is the first *n* uppercase letters in alphabetical order. The second line contains the integer *k*.

Output: The output should contain a single string of *n* uppercase letters representing the *k*-th lexicographical derangement of the starting sequence. Indexing begins at 1 (the first derangement has *k* equal to 1; the second derangement has *k* equal to 2, etc.)

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

Assumptions: *n* will be an integer between 2 and 26, inclusive.
k will be an integer between 1 and 100,000, inclusive.
There will be at least *k* valid derangements for the given *n*.
All input will be valid.

Sample Input #1:

```
4
6
```

Sample Output #1:

```
CDBA
```

Sample Input #2:

```
9
333
```

Sample Output #2:

```
BAECGHDIF
```

Problem 9.4 Not So Boring Contest Scoring (page 1 of 2)

Overview: Sort contestants based on total score and output their ranking.

Description: In an effort to boost its 2% yield rate, the Underwater Basket Weaving Institute of Technology (UBWIT) is holding its first ever basket weaving contest (BWC). Renowned basket weavers from all over the world have left their underground hobbit holes to attend this prestigious event.

As underwater basket weaving is a challenging and dangerous activity, the BWC designed problems of three different levels of difficulty in order to cater to a wide range of aspiring basket weavers and to minimize human casualties. The problems are divided into categories of two-, five-, and nine- point problems, with ranks being decided by highest total score. Ties are broken by the most nine-pointers, followed by the most five-pointers, and then the most two-pointers.

Unfortunately, the even more prestigious Underwater Basket Weaving Polytechnic Institute (UBWPI), angry and confused by the fact that UBWIT is finally taking some initiative, has publicly expressed their intentions to sabotage the contest, so UBWIT must tabulate the results as quickly as possible before UBWPI tinkers with them.

As the only student at UBWIT that is not specializing in basket weaving (and therefore not participating in this competition), the job of stopping UBWPI and writing a program that will calculate the contest results has fallen to you. Best of luck – the fate of UBWIT rests in your hands.

Time Allocation: 1 second

Input: The first line contains an integer *k*, representing the number of teams that are participating in BWC. The next *k* lines will each contain a single-word name *s*, followed by three integers *t f n*, each separated by exactly one space, representing the number of two-, five-, and nine- point problems solved by contestant *s*, respectively.

Output: The output should contain *k* lines of output, with each line containing a single-word contestant name. Print the contestants in decreasing order of performance, starting at first place.

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

Assumptions: *k* will be an integer between 1 and 100, inclusive.
s will be no longer than 50 characters and will contain only the uppercase and lowercase characters `A-Z` and `a-z`.
t, *f*, and *n* will each be an integer between 0 and 5, inclusive.
No two contestants will solve exactly the same number of two-, five-, and nine-pointers. That is, all ties can be broken.
All input will be valid.

Problem 9.4 Not So Boring Contest Scoring

Sample Input #1:

```
5
Coil 3 3 3
Yucca 3 2 4
Twine 4 5 2
Wicker 0 2 5
Split 5 2 3
```

Sample Output #1:

```
Wicker
Yucca
Twine
Coil
Split
```

Sample Input #2:

```
2
Whale 5 5 4
Porcupine 3 4 5
```

Sample Output #2:

```
Porcupine
Whale
```

Problem 9.5 **Matching Pears** (page 1 of 2)

Overview: Compute the maximum number of pears that can be ripened given limited resources.

Description: Omega-Pears is a rather strange company specializing in the lucrative pear-ripening business. Pears ripen upon exposure to ethylene, and Omega-Pears analyzes each pear to determine the optimal range of ethylene exposure.

Through careful testing, Omega-Pears has determined the optimal ripening ranges for each of the *n* pears in its most recent batch. The *i*-th pear will ripen optimally in any ethylene concentration between the a minimum concentration of min_i and the maximum concentration max_i.

The company's laboratory has room for *m* different ripening chambers. The *j*-th chamber has an ethylene concentration of e_j and can hold up to c_j different pears. Pears ripen independently; that is, placing multiple pears into the same chamber will not affect the ripening process of other pears in the same chamber. Each pear can only be placed in one chamber, but not all chambers need be used.

Omega-Pears meet strict deadlines and has time for only one round of pear ripening. Omega-Pears has enlisted your help to match the pears to the available ripening chambers so that the maximum number of pears ripen. Good luck!

Time Allocation: 1 second

Input: The first line contains a single integer *n*, representing the number of pears.

The next *n* lines each contains two integers min_j max_j, separated by exactly one space, representing the minimum and maximum ethylene concentrations allowed for pear *j*.

The next line contains a single integer *m*, representing the number of ripening chambers.

The next *m* lines will each contain two integers e_k c_k, separated by exactly one space, representing the ethylene concentration and pear capacity of ripening chamber *k*.

Output: The output should consist of a single integer *r*, representing the maximum number of pears that can be ripened given the above conditions.

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

Problem 9.5 Matching Pears

Assumptions:

n and *m* will each be an integer between 1 and 2009, inclusive.
min_j will be an integer between 1 and 1000, inclusive.
max_j will be an integer between min_i and 1000, inclusive.
e_k will be an integer between 1 and 1000, inclusive.
c_k will be an integer between 1 and 2009, inclusive.
Multiple pears may have the same min_j and max_j.
No two chambers will have the same e_k.
r will be an integer between 0 and *n*, inclusive.
All input will be valid.

Sample Input #1:

```
3
1 3
2 5
3 4
3
2 3
3 1
4 3
```

Sample Output #1:

```
3
```

Sample Input #2:

```
6
1 2
2 3
3 3
1 2
5 7
6 7
2
4 3
8 2
```

Sample Output #2:

```
0
```