**October 29th, 2011**

# Taylor High School Programming Contest

*"Nunc mane desperationis."*

**General Guidelines:**

- You will have 2 hours to complete the programming portion of the contest.
- There will be 20 problems, with varying point values.
    - There are no point deductions for incorrect solutions; solutions may be submitted multiple times.
- Witchcraft will not be tolerated.

| # | Name | Points |
|---|---|---|
| **1** | Aperture Image Format | 1 |
| **2** | The Bell Tower | 26 |
| **3** | Science without Results Is Just Witchcraft | 3 |
| **4** | You're Out of Your Vector! | 13 |
| **5** | Reverse Engineering | 14 |
| **6** | This Reminds Me of a Puzzle | 3 |
| **7** | Prince of the Red Lions | 10 |
| **8** | Minecraft | 4 |
| **9** | Alice and Bob | 11 |
| **10** | Gruntpocalypse | 5 |
| **11** | Amnesia | 2 |
| **12** | Very Specific Type of Security | 18 |
| **13** | ABXY | 4 |
| **14** | Order of Not-perations | 6 |
| **15** | Chemistry Cat Loves Moles | 7 |
| **16** | Injection Vector | 5 |
| **17** | Class Rosters | 9 |
| **18** | Pathways into Darkness | 7 |
| **19** | Kasiski | 9 |
| **20** | Crossout | 8 |

## Aperture Image Format

**Worth**: 1 point
**Problem File:** pr01.java
**Data File:** None
**Time Allocation**: 1 Second

**General Statement:** As a senior programmer at the Aperture Science Laboratories, Doug Rattmann had the responsibility of printing out ASCII versions of random pictures the boss liked. Unfortunately, Doug accidentally deleted the file for one of the boss' favorite art pieces, which he printed out the night before. Help Doug keep his job by creating an exact duplicate of the ASCII art, as shown below.

**Input:** None

**Output:** Print out the output below exactly as it appears.

**Assumptions:** None

**Sample Input:**
```
None
```

**Sample Output:**
```
********#********
*******###*******
******#####******
*****###*###*****
****###***###****
***###**#**###***
**###*******###**
*###############*
*****************
```

## The Bell Tower

**Worth:** 26 points
**Problem File:** pr02.java
**Data File:** pr02.dat
**Time Allocation**: 10 Seconds

**General Statement:** Challenge mode engaged. While visiting Ecruteak City, Ash decides to visit the Bell Tower and get through to a hidden entrance in the back. Unfortunately, the tower is designed as a giant, multi-story maze. Ash is relying on you to help him get through the maze without getting lost. Since Ash's Pokémon are low on health and he's low on repels, he needs to get through the tower in the least amount of steps possible, traversing on the least number of floor tiles possible. If multiple paths are possible, take the path with the shortest amount of steps.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set that follows consists of a single line that contains an integer, *l*, which represents the number of floors that follow and a second line that contains 2 integers, *r* and *c*, which represent the dimensions of each floor. Each floor will contain a single line with an integer, d, which represents the floor number, then *r* lines of c characters that represent the layout of the floor. Within the layout, the character "`*`" represents a wall that the player cannot traverse through, "`.`" represents floor that the player can walk on, "`U`" represents a staircase going up, "`D`" represents a staircase going down, "`S`" represents the start of the tower, and "`E`" represents the end of the tower. The floors will be input from bottom floor to top floor. See the table below for more information.

| | |
|---|---|
| `*` | Wall that the player cannot traverse through |
| `.` | Floor space that the player can traverse on |
| `U` | A staircase leading to the floor above |
| `D` | A staircase leading to the floor below |
| `S` | The entrance to the tower |
| `E` | The secret exit at the end of the tower |

**Output:** For each data set, if the tower cannot be solved, print out "`I'm stuck!`" If the tower can be solved, print out the number of steps it takes to solve the maze and the minimum number of floors that the player has to traverse through.

**Assumptions:** The dimensions of the tower will not exceed 15 by 15. The number of levels in each tower will not exceed 5. The entrance and the secret exit can be on any floor. There will only be one shortest path to the exit.

**Sample Input:**

```
1
3
9 12
1
************
*E........U*
************
*.....U*...*
*.******.*.*
*......*.*.*
******.*.*.*
*........*S*
************
2
************
*.........D*
*.**********
*.*U*.D*...*
*.*.**.*.*.*
*...*..*.*.*
******.*.*.*
*........*U*
************
3
************
*..........*
*.********.*
*.*D.....*.*
*.*.*.**.*.*
*.*.*..*...*
*.**********
*.........D*
************
```

**Sample Output:**

```
105 3
```

## Science without Results Is Just Witchcraft

**Worth:** 3 points
**Problem File:** pr03.java
**Data File:** pr03.dat
**Time Allocation**: 1 Second

**General Statement:** Cave Johnson, CEO of Aperture Laboratories, produces shower curtains for the government. To ensure that his company acquires the government contract for federal shower curtains, he wants to make sure all of the curtains he produces meet a specific “Shower Curtain Selection Index”, or SCSI. To compute the SCSI, Johnson formulated a secret equation, which has been reproduced below. Given the parameters, compute the SCSI to ensure shower curtain quality for all.

$$S = \frac{x * y}{-z} + z$$

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of 3 nonnegative doubles, *x, y,* and *z,* which are part of the equation as above.

**Output:** Print out the result of Johnson’s equation, *S*. If the result cannot be computed, print out `WITCHCRAFT`

**Assumptions:** The final result, when possible, will be rounded to 3 decimal places when necessary.

**Sample Input:**
```
4
8 1 3
0 0 0
300 9 2
6 15 2
```

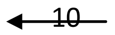
**Sample Output:**
```
0.333
WITCHCRAFT
-1348.000
-43.000
```

## You're Out of Your Vector!

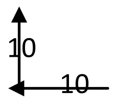
**Worth:** 13 points
**Problem File:** pr04.java
**Data File:** pr04.dat
**Time Allocation**: 1 Second

**General Statement:** Sho Minamimoto, the math freak reaper, finds himself walking down the streets of Shibuya. As he walks, he watches birds fly above him. Being the math nerd that he is, he wants to calculate the vectors of the bird's flight. A vector is composed of two integers, in the form (*x, t*), where *x* is the length of the vector, and *t* is the direction of the vector, in degrees, starting from the positive x-axis and going counter-clockwise. For instance, (10, 90) represents a vector facing North (90 degrees from the positive x -axis) with length 10. Sho wants you to help him add all of the vectors along which the birds travel.

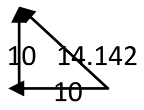
Adding vectors is a simple process. Place the end of each vector at the end of the last vector in a tip to tail fashion. Once you have placed all of the vectors down, draw a vector connecting the tail of the first vector to the tip of the last vector; this vector's length and direction is the final answer. For a visual example, if you have the vectors (10, 180) and (10, 90), then you can draw the vectors out:



The first vector draws as such, 10 at 180 degrees.



Then draw in the second vector 10, at 90 degrees.



Finally, draw in the resultant vector. Measuring the resultant vector, the final answer is (14.142, 135).

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of one line that contains an integer, *z,* followed by *z* vectors in the form "`x y`" which represent real number vectors in the form (*x, t*).

**Output:** For each input, print out the final vector in the form (*x, t*). *t* should be in the range of [0, 359).
**Assumptions:** Answers should be rounded to 3 decimal places. The input and output will never exceed the bounds of the `decimal` structure. You may use any method of vector addition. Vector lengths will never be negative. Angles are given in degrees and will be in the range of [0, 360].

**Sample Input:**
```
2
2
10 180
10 90
3
2 90
6 45
9 30
```

**Sample Output:**
```
(14.142, 135.000)
(16.134, 41.748)
```

## Reverse Engineering

**Worth:** 9 points
**Problem File:** pr05.java
**Data File:** pr05.dat
**Time Allocation**: 1 Second

**General Statement:** Doug, the programmer for the Aperture Image Format, accidentally deleted the program that displays the ASCII art he finds! He still has the files themselves, but he needs to reverse engineer the files to retrieve the program that displays the ASCII images. Luckily for Doug, engineers from previous generations left documentation for the image format. Doug needs you to help him create a program that reads the image format and displays the final image.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of one single line that represents the contents of the image format, which can be interpreted by reading certain offsets, which are sections of the string. The documentation to read the image format is as follows.

The image format is split up into groups of characters called “blocks”; each block consists of a given number of characters, which are known as “bits”. Each of the first 4 blocks will always be 4 bits long. The first block represents the length in bits of each of the blocks after the fourth block. The second block represents the height of the image in the number of characters displayed on the screen. The third block represents the width of the image in the same manner. The fourth block represents the number of blocks after the fourth block. Every block should be read as a separate decimal integer. Every block after the fourth one contains the ASCII value of the appropriate character in decimal format.

**Output:** For each input, print out the final ASCII art image that is read. Print out a blank line after each image.

**Assumptions:** There will always be at least five blocks to each input. The dimensions given by the second and third blocks will match the number of blocks specified by the fourth block. The dimensions of the final image will not exceed 300 characters by 300 characters.

**Sample Input:**
```
3
00040004000400160065006500650065006500650065006500650065006500
6500650065 (Note that this line has been wrapped)
000200010001000166
0003000200020004048048048048
```
**Sample Output:**
```
AAAA
AAAA
AAAA
AAAA

B

00
00
```

## This Reminds Me of a Puzzle

**Worth:** 3 points
**Problem File:** pr06.java
**Data File:** pr06.dat
**Time Allocation**: 1 Second

**General Statement:** While walking through the curious village, Professor Layton encounters puzzles that even he cannot solve. Fortunately, he can purchase hints for the puzzles, after which he can easily solve the puzzles. For each puzzle, he needs three coins to unlock all of the hints. Given how many coins he has, help the professor figure out the number of puzzles to which he can get all three hints.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of a single integer, *c*, that represents how many coins the professor has.

**Output:** For each input, print out the total number of puzzles to which the professor can purchase hints.

**Assumptions:** The number of coins will never be negative.

**Sample Input:**
```
4
2000
6
10
27
```

**Sample Output:**
```
666
2
3
9
```

## Prince of the Red Lions

**Worth:** 10 points
**Problem File:** pr07.java
**Data File:** pr07.dat
**Time Allocation**: 1 Second

**General Statement:** While sailing on the Prince of the Red Lions, Link has gotten lost on the high seas due to the boat's poor navigational skills. Relying on smoke signals sent by a faraway boat, Link must navigate his way out of trouble without the help of his useless ship. Write a program that determines whether Link reaches his destination.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of a single line of smoke signals, where "N" is north, "W" is west, "E" is east, and "S" is south, followed by an 8 by 8 character grid. In the grid, each character represents a certain object on the world map; these character meanings are shown in the table below.

| | |
|---|---|
| * | Open sea the player can travel on |
| . | A rock in the water |
| + | A whirlpool in the water |
| D | Link's destination |
| S | Link's starting point |

**Output:** For each input, determine whether or not the instructions given to Link by the smoke signals would lead him to his destination or result in his demise. Each output should be one line. If Link reaches his destination, output "VICTORY"; if Link hits a rock, output "FAILURE"; if Link is sunk by a whirlpool, output "GAME OVER".

**Assumptions:** The character grid will always be 8 by 8. The smoke signals given will always be valid and will never lead outside of the map area. One of the three situations outlined will always occur.

**Sample Input:**
```
1
NNNNEES
*****.**
********
***.****
********
****+D+*
****+.+*
*.******
***S****
```

**Sample Output:**
```
VICTORY
```

## Minecraft

**Worth:** 4 points
**Problem File:** pr08.java
**Data File:** pr08.dat
**Time Allocation**: 1 Second

**General Statement:** In the world of Minecraft, the objective is to build as much as you can. As a beginning Minecraft player, your task is to build boxes with certain specified heights.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of a single integer, *h*, that represents the height of the boxes. The boxes should consist of only *'s.

**Output:** For each input, print out the completed box. Print a blank line between each box.

**Assumptions:** The output should not be padded with any spaces. The input will be nonnegative.

**Sample Input:**
```
3
1
6
3
```

**Sample Output:**
```
*

******
******
******
******
******
******

***
***
***
```

## Alice and Bob

**Worth:** 11 points
**Problem File:** pr09.java
**Data File:** pr09.dat
**Time Allocation**: 1 Second

**General Statement:** Alice and Bob share a cryptographic connection online to send data to each other. Eve, being the eavesdropper that she is, wants to listen in on their conversation and ambush their date. Alice and Bob know that their connection is secure based on the number of recurring letters existing within their encrypted strings. Write a program that computes the Security Index of their encryption using a formula. To use the formula, you must find the greatest number of occurrences of a single character, and divide it by the number of different characters.

$$S = \frac{g}{t}$$

(where $S$ = Security Index, $g$ = greatest number of occurrences of a single character, $t$ = total different number of characters)

For instance, in the encrypted string "ABCDADCC", $g$ = 3 (because the greatest number of occurrences of a single character, in this case C, is 3) and $t$ = 4 (because there are only 4 unique characters).

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of a single line, which is the encrypted material that Bob has sent to Alice.

**Output:** For each input, print out the Security Index of the encrypted material rounded to the third decimal place.

**Assumptions:** The Security Index is rounded to the nearest thousandth. Always display the index up to the third decimal place and at least a single digit to the left of the decimal. Treat upper and lower case letters as the same letter.

**Sample Input:**
```
2
AAAAAAAAAAAAAAAAA
XXYYZZC
```

**Sample Output:**
```
17.000
0.500
```

## Gruntpocalypse

**Worth:** 5 points
**Problem File:** pr10.java
**Data File:** pr10.dat
**Time Allocation**: 1 Second

**General Statement:** While at CASTLE Base, Dr. Halsey wants to keep track of how many grunts Jun has removed from the grounds. Unfortunately, this data is mixed in with other data, such as how many elites, zealots, and skirmishers he has eradicated as well. Help Dr. Halsey process the data by counting the number of grunts Jun has removed.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of one line that contains a string of characters. Each "+" represents one grunt.

**Output:** For each input, print out the total number of grunts represented in the scoring list.

**Assumptions:** The input will always contain only ASCII values.

**Sample Input:**
```
4
++++
5dsjg++n03++n+g0n+300+ij+3g+
+++++t984h[a-z]||---+
++$**##(++)..
```

**Sample Output:**
```
4
9
6
4
```

## Amnesia

**Worth:** 2 points
**Problem File:** pr11.java
**Data File:** None
**Time Allocation**: 1 Second

**General Statement:** Due to being stranded on Derris Kharlan for far too long, Kratos has forgotten your names! Help Kratos by creating a program that prints out your team number, school name, and team members.

**Input:** None

**Output:** Print out the sample exactly as shown with “`1000`” replaced with your team number, “`Taylor High School`” replaced with your school name, and the names of your team members on separate lines.

**Assumptions:** You will not fight over the order of your team members.

**Sample Input:**
```
None
```

**Sample Output:**
```
1000
Taylor High School
Lloyd Irving
Colette Brunel
Raine Sage
```

## Very Specific Type of Security

**Worth:** 17 points
**Problem File:** pr12.java
**Data File:** pr12.dat
**Time Allocation**: 1 Second

**General Statement:** Dom Cobb wakes up thinking about elephants. Fearful that he has been the victim of someone else's inception, he tries to recall how he began thinking about purple elephants (careless as he is, he has lost his totem). Help Cobb figure out whether he has been the victim of a crime.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set begins with a single line containing an integer, *t*, and a string, *i*, separated by a space. *i* represents Cobb's current thought, while *t* represents the number of ideas Cobb can recall that relate to him reaching the current thought. The next *t* lines will consist of two strings, separated by a space. The first string will represent the idea that moved Cobb to the second string. For instance, if the two strings are "`box cube`", then that means Cobb thought about a `box`, which then led him to think about a `cube`.

If Cobb can trace back the idea to an original inspirational thought, then it was only just a dream. If Cobb's ideas become cyclical, then he has been the victim of an inception heist.

Consider the following example: Cobb traces his thoughts and comes up with the following thought patterns, with his current thought being `pokeball`.

```
box cube
sphere pokeball
cube sphere
```

Cobb simply had a dream because if you trace the ideas from `pokeball` to `sphere`, `sphere` to `cube`, and `cube` to `box`, `box` was not the product of any other thought; therefore, Cobb's thought patterns were not in a cycle and Cobb was not victim to inception.

**Output:** If the thought patterns Cobb has are found to be cyclical, print out `INCEPTION`. If not, then print out `ONLY JUST A DREAM`.

**Assumptions:** None

**Sample Input:**
```
4
3 pokeball
box cube
sphere pokeball
cube sphere
3 tv
pokeball tv
tv pokemon
pokemon pokeball
10 fire
ash coal
bar genius
poke pokemon
pokemon ash
facebook poke
wall facebook
coal fire
steel bar
purple elephant
blue purple
3 a
b a
c b
b c
```

**Sample Output:**
```
ONLY JUST A DREAM
INCEPTION
ONLY JUST A DREAM
INCEPTION
```

## ABXY

**Worth:** 4 points
**Problem File:** pr13.java
**Data File:** pr13.dat
**Time Allocation**: 1 Second

**General Statement:** For generations of gaming consoles, a, b, x, and y have been essential controller buttons. Engineers at the Black Mesa Research Facility want to create a video game and need you to create a program that maps a, b, x, and y to in-game actions. The button mapping can be found in the table below.

| | |
|---|---|
| a | ACTION |
| b | JUMP |
| x | CROUCH |
| y | BARRELROLL |

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of one line that contains a string of characters, `a`, `b`, `x`, or `y`.

**Output:** For each input, print out the translated actions on a single line, delimited by spaces.

**Assumptions:** All input will be valid and uppercase.

**Sample Input:**
```
3
ABXY
BXB
Y
```

**Sample Output:**
```
ACTION JUMP CROUCH BARRELROLL
JUMP CROUCH JUMP
BARRELROLL
```

## Order of Not-perations

**Worth:** 5 points
**Problem File:** pr14.java
**Data File:** pr14.dat
**Time Allocation**: 1 Second

**General Statement:** Catherine and Katherine are having a serious argument; their spat has escalated into a screaming match, with each side yelling statements with absurd amounts of “NOT” prefixed to the beginning of their sentences. Help out poor Vincent, an innocent bystander to this scene, understand what’s going on by removing the extraneous “NOT”’s from their screaming.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of a single line that begins with “YOU ARE”, followed by zero or more “NOT”’s, and a statement.

**Output:** For each input, output the same statement as before, but without the extraneous “NOT”’s.

**Assumptions:** None

**Sample Input:**
```
3
YOU ARE NOT NOT NOT SMART
YOU ARE NOT A GOOD PERSON
YOU ARE NOT NOT NOT NOT NOT NOT A JERK
```

**Sample Output:**
```
YOU ARE NOT SMART
YOU ARE NOT A GOOD PERSON
YOU ARE A JERK
```

## Chemistry Cat Loves Moles

**Worth:** 7 points
**Problem File:** pr15.java
**Data File:** None
**Time Allocation**: 1 Second

**General Statement:** Chemistry cat loves chemical elements, but some of his elements have been scrambled! His periodic table has been drawn on and now every other element has been written backwards. Help chemistry cat write a program to fix the names on his beloved period table of elements.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of a single line that represents a row on the periodic table. Reverse the characters in every other element (starting from the second element).

**Output:** For each data set, print out the row of elements with every other element reversed.

**Assumptions:** There will be at least one element in each row.

**Sample Input:**
```
2
antimony cinesra helium muinaru
plutonium negortin hydrogen
```

**Sample Output:**
```
antimony arsenic helium uranium
plutonium nitrogen hydrogen
```

## Injection Vector

**Worth:** 4 points
**Problem File:** pr16.java
**Data File:** pr16.dat
**Time Allocation**: 1 Second

**General Statement:** MySQL is a common database technology, designed to efficiently store data on a server. Unfortunately, beginner MySQL database programmers can make the rookie mistake of leaving "injection vectors" in their code. An injection vector is code that allows malicious users to obtain database records that are generally private. Bill, being a novice programmer, leaves such a hole in his code. If Steve, being the black hat hacker that he is, wants to he could simply type in some MySQL commands in quotes to obtain complete server access. Write a program for Bill to check input strings for the possibility of MySQL injection. An injection string consists of any string that is prefixed and suffixed by a single quotation mark and does not contain any single quotation marks anywhere else; for instance, `' OR INSERT HAXX'` is an injection string.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set consists of a single string that should be checked for possible injection.

**Output:** For each data set, if the string is an injection string, print out "`HAXX`", else, print out "`PASS`"

**Assumptions:** You are not actually a MySQL programmer; otherwise, my apologies.

**Sample Input:**
```
4
' UNION SELECT * FROM USERS'
' OR 1; DROP DATABASE"
' UPDATE USERS WHERE Q = "1"'
INSERT VALUES INTO USERS'
```

**Sample Output:**
```
HAXX
PASS
HAXX
PASS
```

## Class Rosters

**Worth:** 8 points
**Problem File:** pr17.java
**Data File:** pr17.dat
**Time Allocation**: 1 Second

**General Statement:** Professor Oak is teaching a class on how to be a Pokémon expert. Given the total number of students who will take the class, the number of classes, and the minimum number of students that can be in each class, determine the number of possible different class rosters.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set that follows consists of a single line that contains three integers: *x*, which represents the total number of students, *y*, which represents the number of classes Professor Oak will hold, and *z*, which represents the minimum number of students Professor Oak will put in each class.

**Output:** Output the total number of different rosters Professor Oak can make.

**Assumptions:** There will not be more than 12 students in total.

**Sample Input:**
```
5
10 2 1
12 3 2
7 2 1
12 4 1
9 3 2
```

**Sample Output:**
```
9
28
6
165
10
```

## Pathways into Darkness

**Worth:** 10 points
**Problem File:** pr18.java
**Data File:** pr18.dat
**Time Allocation**: 1 Second

**General Statement:** Joe Staten has found himself lost in enemy territory; he knows that to enter the enemy base secretly, he must follow a zigzag pattern on the ground. Help Joe by writing a program that creates a zigzag pattern for him. The pattern starts at 1 in the top left corner and then zigzags from top to bottom as shown in the samples.

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set will consist of a single integer, *x*, that represents the length and width of the pattern.

**Output:** For each input, print out the spiral that is of *x* length and width. Print a blank line after each output. If some numbers are multiple digits, pad the smaller digits with 0's to align all of the numbers (see the second sample).

**Assumptions:** *x* will be greater than 0 and no greater than 1000.

**Sample Input:**
```
2
2
4
```

**Sample Output:**
```
1 2
4 3

01 02 03 04
08 07 06 05
09 10 11 12
16 15 14 13
```

## Kasiski

**Worth**: 9 points
**Problem File:** pr19.java
**Data File:** pr19.dat
**Time Allocation**: 1 Second

**General Statement:** The Kasiski Test is a method used for attempting to crack a Vigenere Cipher. The method works by attempting to deduce the length of the keyword used in the cipher. To do this, the cipher text is searched for repeat strings. Conventionally, the strings should be greater than 3 in length to root out coincidence and ensure the likelihood of discovering a real word. The distance between the start of two occurrences of a repeat pattern is likely to be a multiple of the keyword length. This is because if the two conditions (repeat occurrences and distance being a multiple of keyword length) are true, the keyword letters would have lined up to the same pattern both times.

Key: **thst**hsthsthst**thst**hsthsthsthsthst
Text: **pink**istoo**pink**formyfinetastes

In other words, the distance must be divisible by the keyword length. (Above, 3 divides 9.)

Your program should receive an encoded string, discover the distances between repeating patterns in the string, and return the factors of all distances in sorted form (lowest to highest).

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set contains a string, *s,* which represents the cipher text material.

**Output:** For each data set, print out the factors of all distances sorted from least to greatest, with a space between each distance. If there are no factors, print "NONE"

**Assumptions:** There will be no more than one recurring string in each data set. Recurring strings will be at least 3 characters long.

**Sample Input:**
```
2
NDYDEAIGDTOFEROFNDYDEAGFLBHM
ABCDEFGHIJK
```

**Sample Output:**
```
1 2 4 8 16
NONE
```

## Crossout

**Worth**: 8 points
**Problem File:** pr20.java
**Data File:** pr20.dat
**Time Allocation**: 1 Second

**General Statement:** While on the hunt for some Moa at Sword Base, Jorge started doodling on a sheet of paper. However, he was ambushed by a group of Covenant and was unable to finish his cross pattern. Write a program that allows Jorge to finish his drawing pattern while he is in combat. His pattern is to take a word, then draw out from the center into four directions: up, down, left and right. For instance, if given the word "given," Jorge would doodle:

```
    N
    E
    V
    I
NEVIGIVEN
    I
    V
    E
    N
```

**Input:** The first line of input will contain a single integer, *n*, that represents the number of data sets to follow. Each data set contains a string, *s,* which represents the word to be used when doodling.

**Output:** For each data set, print out the completed doodle. Print a blank line after each doodle.

**Assumptions:** The string may consist of any ASCII characters.

**Sample Input:**

```
1
TEST
```

**Sample Output:**

```
   T
   S
   E
TSETEST
   E
   S
   T
```