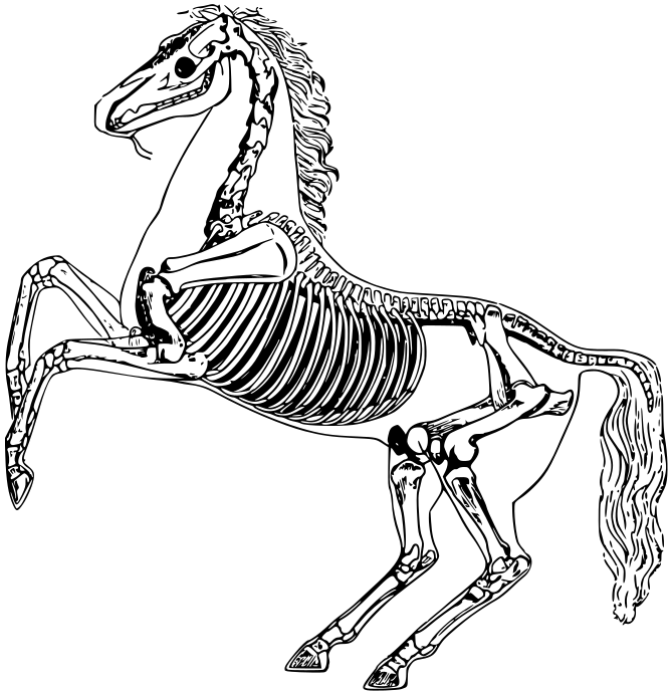
October 28, 2017

Taylor High School

Hands-On Contest



Problems

Number	Title
1	Anagram Candy
2	Base Fun
3	Candy Corn
4	Costume
5	Ghost Battle
6	Ghost Year
7	Haunted House
8	Haunted Mansion
9	Monster Talk
10	Restroom Break
11	Snakes on Plains
12	Spooky Ghosts

1. Anagram Candy

Program Name: Anagram.java **Input File: anagram.in**

Info: Right after trick or treating, Johnny was busy looking at his candy and finding anagrams of the candy names. His mom noticed his interest in anagrams and proposed a game for her son. He had to find if 2 words were anagrams of each other, letters missing from being an anagram, or not anagrams. For each one he got right, he gets another piece of candy. Your job is to write a program for Johnny to solve each puzzle.

Input: The first line will contain an integer, *n*, of the number of data sets. The next *n* lines contain two strings separated by a single space. Strings will only contain capital letters.

Output: If the two strings are anagrams of each other, print "ANAGRAM." If the second string is missing letters to be an anagram of the first string, print out all letters missing even if they repeat with no spaces between and in alphabetical order. For example, if the first string is "HELLO" and the second string is "OH", you would print out "ELL". If there is no way to add letters to the second string to make it an anagram, print out "NOT ANAGRAM". Remember you can only ADD letters, not replace or remove them, to make anagrams.

Sample Input:

```
5
HELLO OEL
POTATO OTTPOA
FRANCE REFAND
ALEX LXEAR
HYDROXYDEOXYCORTICOSTERONES HYDROXYDESOXYCORTICOSTERONE
```

Sample Output:

```
HL
ANAGRAM
NOT ANAGRAM
NOT ANAGRAM
ANAGRAM
```

2. Base Fun

Program Name: BaseFun.java **Input File: basefun.in**

Info: Sam the banker needs help making a simple programming calculator to help with financing his company's huge Halloween Festival party. He needs you to create a program that reads two base-10 numbers and converts them to a requested base and outputs their result in the requested base.

Input: The first line is an integer indicating the number of test cases to follow. Each test case will on one line with 4 data points each separated by a space. The 1st integer indicates the desired base. The 2nd integer indicates the first number in the calculation. The mathematical symbols "+-*/" indicates their respective operations. The 3rd integer indicates the second number in the calculation.

Output: A string in the format as follows: the first number converted to the proper base + " " + Mathematical Operator + " " + second number converted to the proper base + " = " + result converted to the proper base.

Note: If the second number is zero, then just print out "Cannot divide by 0".

Sample Input:

```
4
10 241412 + 324422
16 12121 + 21221
2 1111 - 1111
5 1776 / 0
```

Sample Output:

```
241412 + 324422 = 565834
2f59 + 52e5 = 823e
10001010111 - 10001010111 = 0
Cannot divide by 0
```

3. Candy Corn

Program Name: Corn.java **Input File: N/A**

Info: Halloween is around the corner. Little Billy's favorite treat during the time is Candy Corn, so it's your gift for him this year.

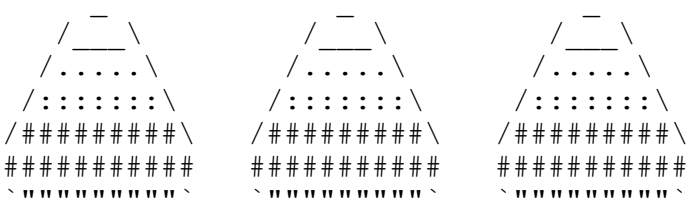
Input: There is no input for this problem.

Output: Print out the candy corns. Each candy corn is separated by three spaces.

Sample Input:

N/A

Sample Output:



4. Costume

Program Name: Costume.java **Input File: costume.in**

Info: Sarah is making her own costume this year. She knows exactly what she needs from the store but doesn't know how much it will cost her, especially since there is a Halloween discount of 15% before tax. She wants to go to the store with the right amount of money including tax to get all her items and wants to include all the discounts. Sarah goes to the store's website online to look up the prices. Your job is to calculate the total price of her items based on the prices including sales tax and discount.

Input: The first line will contain an integer indicating the number of data sets. The next line will contain a number, *n*, that is the inventory count of the store. The next *n* lines are formatted to where it has the item (multiword items are separated by hyphens) and the price separated by a space. After that, there is another integer, *k*, on a separate line that represents the number of different items Sarah needs. The next *k* lines are formatted to where the item and the quantity she needs are separated by a space. In order to qualify for the Halloween discount (15% off) that is applied BEFORE SALES TAX, there needs to be at least 5 distinct items and a total price of at least $25. Sales tax is 8.25%. Note that each data set has its own inventory.

Output: Print the total price with a '$' in front. Round to two decimal places.

Sample Input:

```
1
10
Ribbon 4.25
Sewing-needles 6.00
Brushes 1.50
Paint 2.00
Tape 2.75
Cardboard .50
Styrofoam .50
Glue-bottles 1.75
Wig 5.00
Paper .50
5
Tape 4
Wig 1
Paint 2
Cardboard 12
Glue-bottles 3
```

Sample Output:

```
$28.75
```

5. Ghost Battle

Program Name: GhostBattle.java **Input File: ghostbattle.in**

Info: Frank, a retired fighting ghost, now runs a fight club for ghosts – but you don't know that. Recently, Frank has been betting poorly and has lost lots of money as a result. Frank wants a better way to determine who he should bet on. So, he has hired you to create a fight simulator which will pit two fighters against each other and simulate an outcome.

Input: The first line contains the seed necessary for the Random object you will use to determine the actions of each fighter. The second line contains an integer *n* which denotes the number of fights, or data sets. The next *n* data sets each contain three lines. The first and second lines of each data set contain the names of the fighters, their health points, and their attack value, each separated by a single space. Health points and attack values are integers. The third line of each data set contains the name of the fighter who will go first.

Output: Print out the name of the winning fighter of each fight.

Fight Procedure:

1. The nextBoolean() of the Random object determines the action of the fighter.
    a. If true, the fighter will attack their opponent, deducting their attack value from their opponent's health points.
    b. If false, the fighter will defend against the next attack of their opponent, halving their opponent's next attack regardless of when the next attack may occur. Note that consecutive defensive moves do not stack. For example, the same fighter performing two defensive moves will only halve the opponent's next attack, not the opponent's next two attacks.
2. Turns will alternate between the two fighters until one of the fighter's health points drops to zero or below.

Sample Input:

```
55223
2
Bubbles 209 46
Yankins 198 20
Bubbles
Empyr 196 34
Desir 74 45
Desir
```

Sample Output:

```
Bubbles
Empyr
```

6. Ghost Year

Program Name: GhostYear.java **Input File: ghostyear.in**

Info: Ghost years are revered throughout the ethereal world. During a ghost year, ghosts can manifest in the physical world and wreak havoc. Ghost years occur two years after the most recent leap year and two years before the upcoming leap year. However, most ghosts never participate in a ghost year because they can never figure out when they occur. The Commission for the Well-Being of Ghosts has designated you to write a program which will help ghosts determine when ghost years are.

Input: A series of four-digit integers which signify years, each separated by a line.

Output: Print out the given year followed by "is a leap year", "is a ghost year", or "is neither" if the given year is a leap year, ghost year, or neither a leap year nor ghost year respectively.

Sample Input:

```
2072
1992
2005
1922
1800
2092
1994
2061
1884
1947
```

Sample Output:

```
2072 is a leap year
1992 is a leap year
2005 is neither
1922 is a ghost year
1800 is neither
2092 is a leap year
1994 is a ghost year
2061 is neither
1884 is a leap year
1947 is neither
```

7. Haunted House

Program Name: HauntedHouse.java **Input File: hauntedhouse.in**

Info: Jeff is taking his family on a Halloween trip to the Haunted House. However, his daughter became so scared of the horrors in the house that he is forced to try and lead his family out of there! He needs your help to find the quickest way out of the haunted house's maze. He and his family can only go up, down, left and right through the grid.

Input: The first line is an integer indicating the number of test cases to follow. Each test case will begin with a line of 6 integers, each separated by a space. The 1st integer indicates the number of rows of the grid. The 2nd integer indicates the number of columns of the grid. The 3rd integer indicates the starting row location of Jeff and his family in the grid. The 4th integer indicates the starting column location of Jeff and his family in the grid. The 5th integer indicates the exit row location of the grid. The 6th integer indicates the exit column location of the grid. Below the 6 integers is a grid of 1s and 0s. 1s indicate paths where Jeff and his family can go through, and 0s are impenetrable walls.

Note: If the exit location is a wall, then Jeff and his family are doomed. The starting location will always be an open path, however.

Output: The minimum number of steps needed to exit the Haunted House maze from Jeff's current position, or "AHHHH!!!", if Jeff and his family cannot exit the maze. The initial and final steps are counted.

Sample Input:

```
3
6 7 0 0 5 6
1110111
1011101
1001001
1011101
1000001
1111110
3 3 2 0 0 2
111
110
111
5 5 1 0 3 1
01111
11001
01001
01001
01111
```

Sample Output:

```
AHHHH!!!
5
4
```

8. Haunted Mansion

Program Name: Haunted Mansion.java **Input File: hauntedmansion.in**

Info: Frank and his friends were out exploring the Villas of Dracula, a famous neighborhood known to consist of haunted mansions. All streets in the community have at least one house that hasn't been the site of some strange paranormal incident. While exploring, however, Frank and his friends became lost. He remembers that a house is deemed safe if the digits of its house number add up to the index of the first letter of street name in accordance to the alphabet. So "A" would be 1 and "Z" would be 26. Your job is to help Frank find the safe houses in the Villas of Dracula so he can remain safe.

Input: The first line will contain a single integer *n*, the number of test cases. The first line in each case will contain a single integer *a* representing the number of street addresses in each case.

Output: Print out the address of the house that is deemed safe.

Sample Input:

```
3
2
27352 Spooky Oak Dr.
94800 Ghost Cave Ln.
4
2394 Transylvania St.
2374 Transylvania St.
2992 Vampire Court Dr.
3921 Hollow Brook Ln.
3
93382 Grave Fields Dr.
4927 Witches Walk Ct.
1010 Broomstick Ln.
```

Sample Output:

```
27352 Spooky Oak Dr.
2992 Vampire Court Dr.
1010 Broomstick Ln.
```

9. Monster Talk

Program Name: Monster Talk.java **Input File: monstertalk.in**

Info: Billy the monster loves to constantly scare people on Halloween night. In fact, Billy has written a book on how to be scary in hopes that fellow monsters will pick up his style of scaring. Billy's style, a rather unusual method, involves having to randomly say "BOO!" in the middle of a compound sentence that uses a comma followed by a conjunction ("for", "and", "nor", "but", "or", "why"). While editing his book, Billy realized that the print shop left out the word "BOO!" out of his example sentences. Help Billy find these sentences and fix them if needed.

Input: The first line of input indicates the number of sentences to follow. Each line of input will contain a single sentence that may be either a simple or compound sentence.

Output: If a sentence is compound and makes use of a conjunction, fix the sentence in accordance to Billy's method and the format shown below. If not editable, print out "This sentence is fine as is."

Sample Input:

```
5
Jim loves her, but she doesn't feel the same way back.
One method to keep vampires away is by using garlic and pumpkins.
The beast scared the dog, so he gave him a bone from the grave.
Lola isn't afraid of snakes; she's afraid of spiders.
Billy, the big scary monster, loves to scare people.
```

Sample Output:

```
Jim loves her, "BOO!" but she doesn't feel the same way back.
This sentence is fine as is.
The beast scared the dog, "BOO!" so he gave him a bone from the grave.
This sentence is fine as is.
This sentence is fine as is.
```

10. Restroom Break

Program Name: Restroom.java **Input File: restroom.in**

Info: Pogo the clown loves a good scare. He'll hide in the sewers and creep up on children to give them a good spook. However, nothing scares him more than going to the restroom and seeing that all the available urinals are "blocked"; that is, each empty urinal is next to at least one being used by somebody else. Since he wants to cause mischief, and since he also cares about efficiency because he is a comp sci major when he isn't clowning around, he wants to know the minimum number of clowns he needs to enlist to "block" the restroom.

Input: Each line will contain a sequence of characters (either 'u' or 'i') representing the state of the restroom when Pogo walks in. 'u' represents an empty urinal, and 'i' represents a urinal currently in use.

Output: Print out the minimum number of clowns, including himself, Pogo needs to "block" the restroom.

Sample Input:

```
uuuuu
uiuuu
iuuuu
uuuuuuu
uuuuuii
uuiuuiu
```

Sample Output:

```
2
1
1
3
2
1
```

11. Snakes on Plains

Program Name: Snake.java **Input File: snake.in**

Info: Mr. West is setting up Halloween decorations in his humongous front yard. However, it is dark outside and he is in prime snake territory. Also, due to an eye surgery gone wrong, he can only see two black or white, represented by "*" and "-" in the input, respectively. Therefore, he needs your help to determine if a given set of black pixels forms could be a snake. We define a snake as a single line that can go through a set of black pixels such the line goes through each pixel once and only once and each pixel can only be connected to another directly on top, to the left, to the right, or below it. Basically, any valid shape that can be made by a player in the game "Snake".

Input: The first line will contain an integer, *n*, that represents the number of test cases. Each test case beings with a single line with two integers, *a* and *b*, which represent the number of rows and columns in Mr. West's field of vision. The next *a* lines represent Mr. West's field of vision, with "*" representing a black pixel and "-" representing a white pixel.

Output: Print out "Watch your back!" if ALL the black pixels could be a snake, otherwise print "Don't be paranoid."

Sample Input:

```
3
5 5
-----
--**-
---*-
---*-
---*-
5 5
-----
-***-
-*-*-
****-
---*-
5 5
-----
-****
-*--*
-****
-****
```

Sample Output:

```
Watch your back!
Don't be paranoid.
Watch your back!
```

12. Spooky Ghosts

Program Name: SpookyGhosts.java **Input File: spookyghosts.in**

Info: The Scare Corporation generates electricity for ghost cities by collecting the horror of human beings. Although the company is very effective, internal conflicts between employees resulting from the company's inherently competitive atmosphere have made employee grading challenging for the department of Ghost Resources. GR, being lazy, has asked you to create a program which will rate the performance of employees.

$$Rating = Transparency \times (\frac{Creepiness + Deviousness}{2})$$

Input: A series of lines each containing a ghost's name, transparency, creepiness, and deviousness separated by a single space in that same order. Transparency is a double ranging from 0 to 1, inclusive. Creepiness and deviousness are integer values ranging from 0 to 100, inclusive.

Output: Print out the name of each ghost followed by their rating (rounded to two decimal places), ordered by their ratings from largest to smallest.

Sample Input:

```
Bubbles 0.235 30 53
Yankins 0.800 68 50
Empyr 0.563 41 35
Desir 0.804 81 85
Haunn 0.454 51 94
```

Sample Output:

```
Desir 66.73
Yankins 47.20
Haunn 32.92
Empyr 21.39
Bubbles 9.75
```