# PSCSTA Programming Competition
# December 13th, 2014

# NOVICE DIVISION

- Complete the problems in any order.
- All problems have a value of 60 points.
- Your program should not print extraneous output. Follow the format exactly as given in the problem.
- There is a 5-point penalty for each incorrect submission for problems that are eventually judged correct.
- Time will be used to break point ties.

| **Problems** | **Points** | **Notes** |
|---|---|---|
| Gorf | | |
| Go Salmon | | |
| Pokey | | |
| Power to Luigi | | |
| Red Light Green Light | | |
| Clock-Paper-Scissors | | |
| Sammy Says | | |
| Game Reviews | | |
| Nomopoly | | |
| Tacks | | |
| Bopscotch | | |
| Chaseball | | |
| Invaders | | |
| Card Sort | | |
| Toe Tic Tac | | |

# Good luck!

All problems in this packet were created by Frances Davies.

# Gorf

Input file: gorf.dat

In Gorf, players hit a small, white ball with a Gorf club. The white ball flies through the air towards a hole in the ground. The Gorf player's goal is to hit the ball into the hole. You are to use the quadratic formula (shown below) to predict how far away from the player the ball will land.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The starting location will be given by using the equation above with - before the square root. The location the ball will land is given by using the equation above with + before the square root. To find the distance, you must subtract the starting location from the landing location.

**Input:**
The first line will contain the number of lines to follow. Each line will contain a, b, and c from the quadratic equation. You do not need to check the validity of the input. They will be separated by single spaces.

**Output:**
For each set of a, b, and c, output the ball's distance traveled rounded to the nearest tenth. Each output should be on its own line.

**Example Input:**

```
3
-1 4 0
-1 10 -11
-0.08 5 -50
```

**Example Output:**

```
4.0
7.5
37.5
```

# Go Salmon

Input file: gosalmon.dat

The object of the game Go Salmon is to collect pairs of cards. Two cards make a pair if they are the same number or are both aces, kings, queens, or jacks. Your job is to test whether two cards make a pair or not.

**Input:**
The first line consists of the number of elements in the file. The following lines each contain two cards. A card will be represented by a number from 2 to 10, or by J, Q, K, or A (jack, queen, king, or ace, respectively).

**Output:**
Output PAIR if the cards make a pair. Otherwise output GO SALMON. Each output should be on its own line.

**Example Input:**
4
2 2
7 9
A 10
K K

**Example Output:**
PAIR
GO SALMON
GO SALMON
PAIR

# Pokey

Input file: pokey.dat

In the game of Pokey, players draw 5 cards and place bets based on their hands. Write a program that prints the best hand you can make. Here are the types of hands that exist, in order from worst hand to best hand:

Zilch - No cards are the same and all five cards are not in sequence, meaning there is no Straight.
Pair - Two cards are the same.
Triple - Three or four cards are the same.
Straight - All five cards are in sequence. For example: 8 9 10 J Q.
Full House - Three cards are the same, and the other two cards are the same

Cards do not loop, meaning K A 2 3 4 is not a straight. A is high.

**Input:**
The first line will contain the number of lines of input to follow. The following lines each contain a hand of 5 cards. Each card will be separated by a single space. A card is represented by a number 2 to 10, jack, queen, king, or ace (represented by J, Q, K, or A, respectively).

**Output:**
Output the best type of hand the five cards make up. For example, in every triple there will also be a pair. However, “triple” should be printed because it is a better hand. Each hand should be on its own line.

**Example Input:**

```
4
8 2 6 6 7
7 8 9 10 J
4 4 10 4 10
2 3 5 9 Q
```

**Example Output:**

```
PAIR
STRAIGHT
FULL HOUSE
ZILCH
```

# Power To Luigi

Input file: powertoluigi.dat

Luigi is jealous of Mario for taking the spotlight all the time! Write a program that takes a string and replaces every instance of “Mario” with “Luigi”. Mario will always be capitalized. Mario may be a substring.

**Input:**
The first line will be a number representing the number of lines to follow. Each line after will contain a phrase.

**Output:**
Output the same phrases, each on its own line, with all instances of “Mario” replaced with “Luigi”.

**Example Input:**
3
Mario Kart
So long, eh Bowser?
Super MarioLand in Mario World on Mario  Planet 9000

**Example Output:**
Luigi Kart
So long, eh Bowser?
Super LuigiLand in Luigi World on Luigi  Planet 9000

# Red Light Green Light

Input file: redlightgreenlight.dat

In Red Light Green Light, one person, the leader, stands at the end of a field. The players stand at the other end of the field. When the leader yells “Green light!” the players run towards the leader. When the leader yells “Red light!” the players stop running. Based on the time between “Red Light!” and “Green Light!” and the speed a certain player is running, you must calculate how far that player will go. The relationship between speed, distance, and time is shown below:

$$\text{Speed} = \frac{\text{Distance}}{\text{Time}}$$

In order to calculate distance, you must multiply speed by time.

**Input:**
The first line will contain the number of lines to follow. Each line will contain the time (in seconds) between when “Red light!” was yelled and when “Green light!” was yelled, followed by a certain player’s speed (in feet per second). These will be separated by a single space.

**Output:**
Output the distance traveled (in feet) by each player, rounded to the nearest 100th. Each output should be on its own line.

**Example Input:**
3
10 2
2.2123 5.5
35.5 0

**Example Output:**
20.00
12.17
0.00

# Clock-Paper-Scissors

Input file: clockpaperscissors.dat

In a match of Clock-Paper-Scissors, two players each pick either clock, scissors, or paper, and then compare their choices. The game is played two-out-of-three, which means a player must win at least two out of three matches to win the whole game. Write a program to determine the winner of a game.

**Input:**
The first line will contain a number indicating how many games are being played. Each game will be represented by three numbers. A 1 indicates that Player 1 wins that match, while a 0 indicates that Player 2 wins that match. Numbers will be separated by a single space.

**Output:**
For each game, output "Player 1" if Player 1 wins or "Player 2" if Player 2 wins. Each output should be on its own line.

**Example Input:**
3
1 1 0
0 1 0
1 1 1

**Example Output:**
Player 1
Player 2
Player 1

# Sammy Says

Input file: sammysays.dat

In the game Sammy Says, Sammy is telling you what to do. However, you should only do what he says if his sentence starts with “Sammy says ”.

**Input:**
The first line will be a number indicating how many commands Sammy will give you. Each line after will contain a command.

**Output:**
If a command starts with “Sammy says” output the command that follows. Each output should be on its own line.

**Example Input:**

```
5
Sammy says walk
Sammy says jump
Take three steps
Sammy commands you to do a cartwheel
Sammy says crawl forward
```

**Example Output:**

```
walk
jump
crawl forward
```

# Game Reviews

Input file: gamereviews.dat

Before buying a board game or video game, you should always check its reviews online. In this problem, you will process a list of game reviews and output the average star rating that each game received.

**Input:**
The first line will be a number indicating how many reviews need to be processed. Each review will contain the name of the reviewer, the name of the game, and the rating, in that order. These will be separated by commas and single spaces. The name of the reviewer will always be one word.

**Output:**
For each game, output (in alphabetical order based on game name) "_____ gets __ stars" where the first blank is replaced by the name of the game, and the second blank is replaced by the average number of stars the game received, rounded to the nearest 10th.

**Example Input:**

```
10
Alex, Super Luigi Planet, 4
Bob, Nomopoly, 2
Carly, Pac-Dude, 5
Doris, Settlers of Catan, 5
Bob, Super Luigi Planet, 3
Doris, Nomopoly, 5
Ernie, Pac-Dude, 1
Ernie, Nomopoly, 3
Alex, Pac-Dude, 5
Bob, Pac-Dude, 4
```

**Example Output:**

```
Monopoly gets 3.3 stars
Pac-Dude gets 3.7 stars
Settlers of Catan gets 5.0 stars
Super Luigi Planet gets 3.5 stars
```

# Nomopoly

Input file: nomopoly.dat

In Nomopoly, you acquire properties. If another player lands on a property you own, they have to pay you money based on how many houses you have built on that property. For each house on the property, the amount of money a player must pay if they land on that property increases by 125%. Write a program to figure out how much a player would owe you if they landed on your property. Because there are no coins in Nomopoly, after calculating the rent, your program should round *down* to the nearest dollar.

**Input:**
The first line will be a number indicating how many properties you will examine. Then, each property will take up three lines. The first of these three lines will be the name of the property; the second line will be the original rent of the property, when no houses were built, in the format "$xx.xx"; and the third line will be the number of houses on the property now.

**Output:**
Output "If someone lands on _____, they will owe me $_____." where the first blank is the name of the property and the second blank is the rent, rounded *down* to the nearest dollar. Each output should be on its own line.

**Example Input:**

```
2
Atlantis Ave
$30.00
2
Engelberg Way
$58.00
1
```

**Example Output:**

```
If someone lands on Atlantis Ave, they will owe me $151.
If someone lands on Engelberg Way, they will owe me $130.
```

# Tacks

Input file: tacks.dat

To play Tacks, players bounce a rubber ball and try to pick up as many tacks as they can while still being able to catch the ball before it lands. The player who picks up the most tacks and then catches the ball wins.

**Input:**
The first line will contain the number of players participating in the game. The following lines will consist of a player's name, followed by the number of tacks that player was able to pick up. These will be separated by a single space.

**Output:**
Print "____ wins with ____ tacks." where the first blank is the winner's name and the second blank is the number of tacks the winner was able to pick up. In the case of a tie, the person with the alphabetically first name wins.

**Example Input:**

```
5
Sara 3
Bill 3
Lenny 5
Jack 6
Barney 4
```

**Example Output:**

```
Jack wins with 6 tacks.
```

# Bopscotch

Input file: bopscotch.dat

In Bopscotch, players hop across a chalk pattern, then return to their starting point. Your job is to write a program that draws the pattern.

**Input:**
The first line will contain the number of patterns to draw. The following lines will each contain a number indicating the number of steps in that pattern. The number of steps will never be less than 2 or greater than 100.

**Output:**
Output the Bopscotch pattern. Numbers should go from the bottom up and from left to right. Interior lines (lines that are not the top or the bottom) should alternate between containing one number and containing two numbers. In some cases, when there are not enough numbers left at the upper interior line to make a pair, the upper two interior lines will each contain one number. Patterns should be separated by a newline.

**Example Input:**

```
5
2
5
9
10
11
```

**Example Output (see next page):**

**Example Output:**

```
/ 2 \
\ 1 /

/ 5 \
[3][4]
 [2]
\ 1 /

/ 9 \
 [8]
[6][7]
 [5]
[3][4]
 [2]
\ 1 /

/ 10 \
 [9]
 [8]
[6][7]
 [5]
[3][4]
 [2]
\ 1 /

/ 11 \
[9][10]
 [8]
[6][7]
 [5]
[3][4]
 [2]
\ 1 /
```

# Chaseball

Input file: chaseball.dat

In Chaseball, players hit a ball with a bat and then run around a diamond-shaped field. Your job is to write a program that draws the field.

**Input:**
The first line will contain the number of fields to draw. The following lines will each contain a number indicating the number of slashes that make up one side of that field. Each number will never be less than 1 or greater than 50. Fields should be separated by a newline.

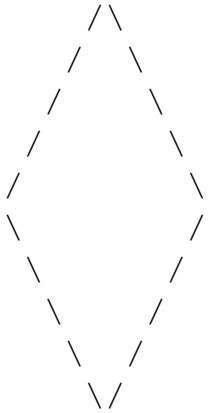
**Output:**
The field.

**Example Input:**

```
2
2
5
```

**Example Output:**

```
 /\
/  \
\  /
 \/
```



# Invaders

Input file: invaders.dat

In Invaders, you control an alien race whose mission is to take over earth. At the beginning of the game, the aliens are confined to one place and spread outwards (in all directions) at a rate of 1000 miles per day. Each day, for every new city the aliens reach, their spread rate doubles. This new rate comes into effect at the beginning of the following day. For example, if the aliens reach two cities in a single day, their spread rate will have doubled twice at the beginning of the next day. The spread rate will never be greater than 250000. You will record the progress of the aliens' takeover.

**Input:**
The first line will contain the number of cities that exist in the game. Each city will be represented by a name, followed by a single space, followed by a colon, followed by another single space, followed by that city's distance from the aliens' starting location.

**Output:**
Each day, you will print the day number, the current spread rate of the aliens, and a message for each city they were able to reach in the order that they were reached. Each message will read "____ has fallen!" where the blank is the name of the city reached. City names will not include spaces or colons. The current spread rate and city messages should each be preceded three spaces, so they appear indented. When the aliens have taken over every city, print "The aliens have taken over!" This message should also be preceded by three spaces.

**Example Input:**

```
5
NYC : 500
Auburn : 3000
Boise : 6770
Seattle : 3100
Sydney : 30000
```

**Example Output (see next page):**

**Example Output:**

```
Day 1
   Rate: 1000 Miles Per Day
   NYC has fallen!
Day 2
   Rate: 2000 Miles Per Day
   Auburn has fallen!
Day 3
   Rate: 4000 Miles Per Day
   Seattle has fallen!
   Boise has fallen!
Day 4
   Rate: 16000 Miles Per Day
Day 5
   Rate: 16000 Miles Per Day
   Sydney has fallen!
   The aliens have taken over!
```

# Card Sort

Input file: cardsort.dat

In some card games, it is useful to sort a hand of cards by suit and number. You will write a program to help with sorting.

**Input:**
The first line will contain the number of hands to follow. Each subsequent line will contain a hand of cards. Each hand will consist of some number of cards, separated by commas and single spaces. Each card is represented by a suit (heart, diamond, spade, or club) followed by a number 2 to 10, jack, queen, king, or ace (represented by J, Q, K, or A, respectively).

**Output:**
Sort each hand by suit (in alphabetical order) and then by number (in the order: ace, 2, … 10, jack, queen, king). Each output should be on its own line.

**Example Input:**

```
2
heart 3, club 10, club K
club 7, diamond 7, spade 7, heart Q, diamond 2
```

**Example Output:**

```
club 10, club K, heart 3
diamond 2, diamond 7, club 7, heart Q, spade 7
```

# Toe Tic Tac

Input file: none

Toe Tic Tac is a game played by two people on a 3 by 3 board. The object of the game is to draw three O's or three X's in a row. This game is so simple that it often ends in a tie.

**Input:**
No input.

**Output:**
Display the tied Toe Tic Tac board exactly as shown below.

**Example Output:**

```
\/ * \/ * /\
/\ * /\ * \/
**********
/\ * \/ * \/
\/ * /\ * /\
**********
\/ * /\ * /\
/\ * \/ * \/
```