Code Warrior Introduction

Problem #1	Novice	1 Point

C programmers: your program name must be: prob01.exe
JAVA programmers: your program name must be: Prob01.class

Task Description

Computer programmers must learn some rudimentary social skills, even if their interpersonal interaction is constrained by digital media. Please write a program to introduce your team to the judges.

Program Input

Your program should print a prompt for the judge to enter his or her name like the **bold face type** below. Then it should read the name from the keyboard input. You may assume that the judge will enter one name only, so you don't have to worry about spaces in the input.

Please enter your name: Joe

Program Output

After reading the judge's name, your program must complete the introduction by (1) greeting the judge, and (2) displaying the name of your school, your mascot, and the names of each team member. You may use the format below as an example:

Hello, Joe!

We are Athens High School Lions: Mike Davis, Anh Nguyen, and Isok Patel

Leap Year

Problem #2	Novice	2 Points

C programmers: your program name must be: prob02.exe
JAVA programmers: your program name must be Prob02.class

Task Description

The ancient race of Gulamatu is very advanced in their calculation scheme. They understand what a leap year is (a year that is divisible by 4 and not divisible by 100 with the exception that years divisible by 400 are also leap years) and they also have similar festival years. One is the Huluculu festival (happens on years divisible by 15) and the Bulukulu festival (happens on years divisible by 55 provided that is also a leap year).

Given a year you will have to state what properties the year has. If the year is neither a leap year nor a festival year, then print a line that the year is an ordinary year. The order of printing the properties (if present) is leap year, Huluculu, Bulukulu.

Program Input

Your program should print a prompt the judge to enter a year like the **bold face type** below. All input is via the keyboard.

Please enter a year: 2000

Program Output

Output the different properties for the year on different lines according to previous description and sample output. Note that there are four different properties.

2000 is a leap year.

Sample Input/Output

Please enter a year: 2000
2000 is a leap year.

Please enter a year: 3600
3600 is a leap year.
36000 is a Huluculu festival year.

Please enter a year: 4515
4515 is Huluculu festival year.

Please enter a year: 2001
2001 is an ordinary year.

Alien Income Tax

Problem #3	Novice	2 Points

C programmers: your program name must be: prob03.exe
JAVA programmers: your program name must be Prob03.class

Task Description

Somewhere in the galaxy there exists a planet where the tax laws are much simpler than on our planet. Each alien receives a personal allowance, plus an allowance for each of its children. Write a program to compute income tax on this alien world. The program must compute taxable income and tax owed according to these formulas:

$$N = I - P - M * C$$
$$X = N / 3$$

The alien's annual income is I, P is the personal allowance of 5000 *dolari* (the alien currency), M is the number of the alien's children, and C is the child allowance of 1000 *dolari*. If the taxable income (N) is negative, then it should be set to zero. The tax owed by the alien is X.

Program Input

Your program should print a prompt the judge to enter the total income and number of children like the **bold face type** below. All input is via the keyboard.

Total income: 11000
Number of children: 4

Program Output

On separate lines, print and label the alien's income, taxable income, tax owed, and income after tax. Follow this format:

Income: 11000
Taxable: 2000
Tax: 666
After Tax: 10334

Time of Day

Problem #4	Novice	3 Points

C programmers: your program name must be: prob04.exe
JAVA programmers: your program name must be Prob04.class

Task Description

Computers and humans sometimes prefer information in different formats. For example, computers often work with time in single values that represent milliseconds or seconds, while humans (sometimes) prefer hours, minutes, and seconds, followed by AM or PM where applicable. Write a program to convert the number of seconds to a human readable format.

Program Input

The program must display prompts for two values on the input line: a start time and a stop time. Each value represents the number of seconds since midnight.

Start time: 3101
Stop time: 81954

Program Output

The program must write three output lines: the start time, stop time, and elapsed time. The start and stop time must be formatted as HR:MIN:SEC AM/PM, and the elapsed time just as HR:MIN:SEC. Remember that 0HR is 12AM!

12:51:41 AM
10:45:54 PM
21:54:13

Check Please?

Problem #5	Novice	5 Points

C programmers: your program name must be: prob05.exe
JAVA programmers: your program name must be Prob05.class

Task Description

The menu for the local Lone Star Diner is as follows:

COFFEE – $0.50
SODA - $1.00
MILKSHAKE - $2.13
PANCAKES - $3.25
WAFFLES - $3.92
HAMBURGER – $4.01
PASTA - $4.44
HOTDOG - $2.98
CHIPS - $0.65
FRIES - $0.99

Write a program that takes a given list of menu items that have been ordered and displays the total bill.

Program Input

The customer's order is contained in am input file (c:\codeware\prob05.in). The input file will contain a list of items ordered by the customer. Each item will be on a separate line. "END" will signify the end of the order. You can assume that all of the items in the order will be spelled correctly.

COFFEE
COFFEE
WAFFLES
PASTA
END

Program Output

The total cost of the order, as follows:

Your total is $9.36

What's the Median?

Problem #6	Novice / Advanced	5 Points

C programmers: your program name must be: prob06.exe
JAVA programmers: your program name must be Prob06.class

Task Description

Median plays an important role in the world of statistics. By definition, it is a value which divides an array into two equal parts. In this problem you are to determine the current median of some long integers.

Suppose, we have five numbers {1,3,6,2,7}. In this case, 3 is the median as it has exactly two numbers on its each side. {1,2} and {6,7}.

If there are even number of values like {1,3,6,2,7,8}, only one value cannot split this array into equal two parts, so we consider the average of the middle values {3,6}. Thus, the median will be (3+6)/2 = 4.5. In this problem, you have to print only the integer part, not the fractional. As a result, according to this problem, the median will be 4!

Program Input

The input file (c:\codewars\prob06.in) contains a number of integers X (0 <= X < 2^31) and total number of integers N is less than 10000. The input file will contain a '0" to signify the end of the list of integers. When you read in a '0', your program must exit without calculating a new median with the number 0.

1
3
4
60
70
50
2
0

Program Output

As each number is read in, recalculate the median and display it to the screen.

1
2
3
3
4
27
4

Poker anyone?

Problem #7	Novice / Advanced	7 Points

C programmers: your program name must be: prob07.exe
JAVA programmers: your program name must be Prob07.class

Task Description

A poker deck contains **52** cards - each card has a suit which is one of clubs, diamonds, hearts, or spades (denoted **C**, **D**, **H**, and **S** in the input data). Each card also has a value which is one of **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**, **jack**, **queen**, **king**, **ace** (denoted **2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A**). For scoring purposes, the suits are unordered while the values are ordered as given above, with 2 being the lowest and ace the highest value.

A poker hand consists of **5** cards dealt from the deck. Poker hands are ranked by the following partial order from lowest to highest

High Card: Hands which do not fit any higher category are ranked by the value of their highest card. If the highest cards have the same value, the hands are ranked by the next highest, and so on.

Pair: 2 of the **5** cards in the hand have the same value. Hands which both contain a pair are ranked by the value of the cards forming the pair. If these values are the same, the hands are ranked by the values of the cards not forming the pair, in decreasing order.

Two Pairs: The hand contains **2** different pairs. Hands which both contain **2** pairs are ranked by the value of their highest pair. Hands with the same highest pair are ranked by the value of their other pair. If these values are the same the hands are ranked by the value of the remaining card.

Three of a Kind: Three of the cards in the hand have the same value. Hands which both contain three of a kind are ranked by the value of the **3** cards.

Straight: Hand contains **5** cards with consecutive values. Hands which both contain a straight are ranked by their highest card.

Flush: Hand contains **5** cards of the same suit. Hands which are both flushes are ranked using the rules for High Card.

Full House: 3 cards of the same value, with the remaining **2** cards forming a pair. Ranked by the value of the **3** cards.

Four of a kind: 4 cards with the same value. Ranked by the value of the **4** cards.

Straight flush: 5 cards of the same suit with consecutive values. Ranked by the highest card in the hand.

Your job is to compare several pairs of poker hands and to indicate which, if either, has a higher

Program Input

The input file (c:\codewars\prob07.in) contains 4 poker hands, each containing the designation of **10** cards: the first **5** cards are the hand for the player named **"Chris Moneymaker"** and the next **5** cards are the hand for the player named **"Phil Ivey".** You can assume that the cards in each players hand are valid cards and that there is only 1 card of each type (i.e. there is only 1 Ace of Spades). You can also assume that the judge's data for this problem will also contain 4 hands of poker (with different cards in each hand).

2H 2D 5S 5C 5D 2C 2S 4S 8C AH
2H 4S 4C 3D 4H 2S 8S AS QS 3S
2H 2D 5S 9C KD 2C 2S 4S 9C KH
2H 3D 5S 9C KD 2D 3H 5C 9S KH

Program Output
Output to the screen the best hand for each player and the player with winner of the hand (all on separate lines). Hands should be described as follows:

2H 3C 4D 5S 7H: High Card {6}

2H 2C 4D 5S 6H: Pair {2s}

2H 2C 4D 4S 6H: Two Pair {4s and 2s} (NOTE: Higher ranking pair is listed first)

2H 2C 2D 5S 6H: Three of a Kind {2s}

2H 3C 4D 5S 6H: Straight {6 High}

2H 4H 9H 5H 6H: Flush {9 High}

2H 2C 2D 4S 4H: Full House {2s full of 4s} (NOTE: <three of kind> full of <pair>)

2H 2C 2D 2S 6H: Four of a Kind {2s}

7H 8H 9H 10H JH: Straight Flush {J High}

If a player's hand ranks the same as his opponent (e.g. both players have a pair of two's) include additional cards needed until the tie is broken or there are no more cards (a tie).

Using the sample input above, your program should output:

Chris Moneymaker: Full House {5s full of 2s}
Phil Ivey: Pair {2s}
Chris Moneymaker wins.

Chris Moneymaker: Three of a Kind {4s}
Phil Ivey: Flush {A High}
Phil Ivey wins.

Chris Moneymaker: Pair {2s}{K-9-5}
Phil Ivey: Pair {2s}{K-9-4}
Chris Moneymaker wins.

Chris Moneymaker: High Card {K}{9-5-3-2}
Phil Ivey: High Card {K}{9-5-3-2}
This hand is a tie.

CSI Crime Lab

Problem #8 | **Novice / Advanced** | **8 Points**

C programmers: your program name must be: prob08.exe
JAVA programmers: your program name must be Prob08.class

Task Description

Your task is to write a program that will help the Houston CSI team accurately solve crimes based on a given set of DNA evidence. You will be given DNA sequence samples from several suspects as well as partial DNA sequence samples that were gathered from a crime scene as input. Your program must decide which suspect could be considered guilty, based on a matching of the DNA sequences.

Program Input

A text file (c:\codewars\prob08.in) containing the names of suspects and crime scene locations followed by a colon, and then the partial DNA sequence associated with that suspect or crime scene locale. The list of suspects will be headed by the line "<Suspects>" and ends with the line "<Scene>", which signifies the start of the crime scene data. <END> will signify the end of the input file.

<Suspects>
Larry King: GACTAATAACTTCATATATACACAGGTTAC
Paula Abdul: GACTATTCATCATAGATAGACAGTACCTAA
Charlie McCarthy: GATTCATTGACATACATACATTAGAGTTCA
<Scene>
On revolver: GACTATTC, GACTAATA
On door: GACTAAT, CATAGAT
On phone: CATACATT, ATTAGAG, ATAGATAG
<END>

Program Output

Only one suspect should be selected from the input suspects list. If no positive match is found, your program should output "NO MATCH". In order to have a positive match, a suspect must have a DNA match found at each of the crime scene locales listed under the <Scene> heading. There may be more than one partial DNA sample sequence obtained from each crime scene locale and they will all be listed, separated by commas.

The suspect is: Paul Abdul

Pizza Ordering System

Problem #9 | **Novice / Advanced** | **9 Points**

C programmers: your program name must be: prob09.exe
JAVA programmers: your program name must be Prob09.class

Task Description

Your employer, Domino's Pizza, has recently decided that it would take pizza orders over the web. The central ordering system will process the web orders and send them as PML documents (Pizza Markup Language) over the company intranet to the appropriate local store based on the customer's address. When the local store receives a new order, an order ticket is printed and picked up by the employee that will make the pizza(s).

PML documents are text documents containing structured information (a pizza order) and are very similar to XML (eXtensible Markup Language) documents. A PML document consists of a series of elements which describe the data enclosed by the elements start and end tags, like this:

{element_name}This is some data{\element_name}

Elements can also have attributes that help distinguish it from other instances of the same element. Element attributes are contained in the element start tag, like this:

{note ID="3343"}This is a note{\note}

The note element has an attribute, "ID" with a value of "3343".

Unlike XML, PML tags and their order are predefined and cannot change. All PML documents contain a single **order** element which has a *number* attribute which defines a unique number for that order. Within the **order** element will be a series of up to 24 **pizza** elements. Each **pizza** element also contains a *number* attribute. The first pizza will be number "1", the second will be number "2", and so on. Each **pizza** is described by an additional three elements—**size, crust,** and **type**. The **type** element contains the pizza description, such as "Hawaiian", "Chicken Fajita", or "custom". If it is a "custom" pizza, up to three (3) **toppings** elements will follow. Each **topping** element will have an *area* attribute with a value of 0, 1, or 2 corresponding to "whole pizza", "first-half", and "second-half" respectively. Lastly, under each **topping** element is a list of up to 12 **item** elements for each topping.

While PML is just structured text, unfortunately, it isn't easy for the Domino's employees to read quickly. Thus, you've been asked to write some code to convert the PML orders into a format that is easier for the employees to read. See the sample Program Output for proper format.

Program Input

The customer order is in c:\codewars\prob09.in and has the following format:

```
{order number="123"}
        {pizza number="1"}
                {size}large{\size}
                {crust}hand-tossed{\crust}
```

```
                {type}custom{\type}
                {toppings area="0"}
                        {item}pepperoni{\item}
                        {item}extra cheese{\item}
                {\toppings}
                {toppings area="1"}
                        {item}sausage{\item}
                {\toppings}
                {toppings area="2"}
                        {item}mushrooms{\item}
                {\toppings}
        {\pizza}
        {pizza number="2"}
                {size}medium{\size}
                {crust}deep dish{\crust}
                {type}pepperoni feast{\type}
        {\pizza}
{\order}
```

Program Output

Output to the screen the customer order as follows:

```
Order 123:
        Pizza 1 – large, hand-tossed, custom
                Toppings Whole:
                        pepperoni
                        extra cheese
                Toppings First-Half:
                        sausage
                Toppings Second-Half:
                        mushrooms

        Pizza 2 – medium, deep dish, pepperoni feast
```

Disco Dilemma

Problem #10 **Novice / Advanced** **11 Points**

C programmers: your program name must be: prob10.exe
JAVA programmers: your program name must be prob10.class

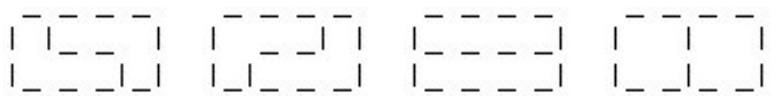
Task Description

Disco Stu has purchased some colorful floor panels for the dance floor of his club. Every Friday his employees will rearrange the panels so the disco dance floor will have a new and exciting look. He has purchased a large supply of panels with seven different shapes. Each panel consists of four 1-foot squares as depicted below:

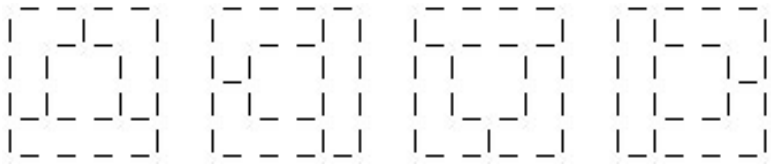
1 2 3 4 5 6 7

Disco Stu needs to know if his dance floor can be completely covered using these panels. He also needs to know how many different patterns his employees can create on the dance floor. The panels can be rotated in 90-degree increments, they cannot overlap, they cannot extend past the edge of the dance floor, and they must cover the dance floor completely.

Your first job is to determine whether the dance floor can be completely covered. Your second job is to determine the number of different patterns that can be created by rearranging the panels. Patterns are considered distinct if the borders between panels are different. For example, for a 4-foot by 2-foot dance floor, there are 4 distinct patterns:



Rotations of patterns are also considered distinct. For example, the following four patterns are distinct:



Program Input

Your program must prompt the judge to enter the length and width of the dance floor in feet like the **bold face type** below. All input is via the keyboard. You can assume that the dance floor will not be larger than 36 square feet. A value of -99 for the length or width measurement should terminate the program.

Enter the length of the dance floor: 2
Enter the width of the dance floor: 4

Program Output

Your program must return the number of distinct patterns that completely cover a dance floor of the given size. If the floor cannot be completely covered, your program must return zero.

There are 4 different patterns that exist for a 2 x 4 dance floor.

Sample Input/Output

Enter the length of the dance floor: 2
Enter the width of the dance floor: 4
There are 4 different patterns that exist for a 2 x 4 dance floor.

Enter the length of the dance floor: 3
Enter the width of the dance floor: 3
There are 0 different patterns that exist for a 3 x 3 dance floor.

Enter the length of the dance floor: 3
Enter the width of the dance floor: 4
There are 23 different patterns that exist for a 3 x 4 dance floor.

Check Please?

Problem #11	Advanced	9 Points

C programmers: your program name must be: prob11.exe
JAVA programmers: your program name must be Prob11.class

Task Description

The menu for the local Lone Star Diner is as follows:

COFFEE – $0.50
SODA - $1.00
MILKSHAKE - $2.13
PANCAKES - $3.25
WAFFLES - $3.92
HAMBURGER – $4.01
PASTA - $4.44
HOTDOG - $2.98
CHIPS - $0.65
FRIES - $0.99

Write a program that takes a given list of menu items that have been ordered and displays the total bill.

Program Input

The customer's order is contained in an input file (c:\codewars\prob11.in). The input file will contain a list of items ordered by the customer. Each item will be on a separate line. "END" will signify the end of the order.

COFFEE
COFFEE
WAFFLES
PASTA
END

Oh, and by the way, when the diner gets really busy, the waiters get a little sloppy with the order entry so there may be a typo (or not) in each line of the order. A typo will only consist of a single missing letter or having a single wrong letter per each item name, but not both. For example:

CONFEE
OFFEE
WAIFLES
PASTE
END

You can assume END will always be spelled correctly.

Program Output

The total cost of the order, as follows:
Your total is $9.36

Mars Rover

Problem #12 | **Advanced** | **12 Points**

C programmers: your program name must be: prob12.exe
JAVA programmers: your program name must be Prob12.class

Task (problem) Description:
This year NASA succeeded in landing two robotic rovers on the surface of Mars. Your team is assigned to write simple navigation code for a Mars Planetary Rover. The rover must be able to execute these simple commands: MOVE, RIGHT, LEFT, DRILL, and END. The rover must supply feedback status to mission control after each DRILL command. A two dimensional grid of ASCII text characters (summarized in *Table 1*) represents the Martian surface. An overlapping (or, perhaps, underlapping) grid of the characters (see *Table 2*) represents the soil and rock composition.

Table 1. Commands

Command	Intent
MOVE N	Move N spaces forward. *Rover must not enter a space if it is an obstruction or crater.*
LEFT	Turn 90° right
RIGHT	Turn 90° left
DRILL	Drill sample from space **in front of** the rover
END	End of current command sequence.

Table 1. Geography

Symbol	Meaning
+	Level surface
X	Obstruction
0	Crater

Table 2. Geology

Symbol	Meaning
H	Hematite
Q	Quartz
F	Feldspar
A	Anatase

Program Input:
The input for this problem can be found in c:\codewars\prob12.in. The number N on the first line indicates the size of the input grid. The next N lines contain N symbols each, in a grid format, separated by spaces. This grid represents the landscape geography. The rover itself is represented by one of the symbols <, >, v, or ^, indicating not only the location, but also the orientation of the vehicle. The next N lines represent the dominant geologic composition of any sample taken from the grid. Each line thereafter is a command, which the rover must execute in sequence. Notice that with a MOVE command, the rover must move forward until (a) it has moved N spaces, or (b) the next space is a crater or an obstacle. *Under no circumstances should the rover move into a space containing a crater or an*

obstacle. Your program must track, at all times, the location and orientation of the Mars rover.

```
10
+ + + + + + + + + +
+ + + + + + X + + +
+ + + O + + X X + +
+ + + + + + X X + +
+ + + + + + + X + +
X + + + + + + + + +
X + + + + + + + + +
+ X + + + O + < + +
+ X + + + + + + + +
+ + + + + + + + + +
F F F A F F F A A F
F F F A F F A F F F
A F F Q F F Q Q F F
F H F F A F F Q F F
F H H H F F A F F F
Q F H H H H F H H A
Q F F H H H H H F F
F Q F F H Q H A F F
A Q A F F A F F F A
F F F F A F A F F F
MOVE 1
DRILL
RIGHT
MOVE 5
RIGHT
DRILL
LEFT
LEFT
MOVE 3
DRILL
END
```

Program Output:

Each time the rover receives a DRILL command, the program must print the dominant mineral of **the space in front of** the rover (not the space in which the rover is currently located). Your program must correctly print each mineral in the correct sequence. No partial credit.

```
ANATASE
QUARTZ
FELDSPAR
```

Word Search

Problem #13 | **Advanced** | **14 Points**

C programmers: your program name must be: prob13.exe
JAVA programmers: your program name must be Prob13.class

Task (problem) Description:

The English teacher promised ***five bonus points*** to the final grade of the student who could find all the hidden words first. Use your programming skills to score some Code Wars points at the same time! Words may appear horizontal, vertical, or diagonal, forward or backward

Program Input:

The first line of the input file (c:\codewars\prob13.in) indicates the number of words, then the number of columns and rows in the puzzle. All values range from four to twenty, inclusive. Each search word appears on a separate line, followed by the puzzle itself.

```
4 8 6
LOOP
FILE
INPUT
OUTPUT
I T I T T T L N
I E O U T P U T
E E P O O P U E
L N O O U O E O
I P L N L U T F
F T T T P U N T
```

Program Output:

Print the puzzle using only the letters that form the words. Print periods in all other locations to maintain proper spacing.

```
. . . . T . . .
. . O U T P U T
E . P . O . . .
L N . O . . . .
I . L . . . . .
F . . . . . . .
```

March Madness

Problem #14	Advanced	18 Points

C programmers: your program name must be: prob14.exe

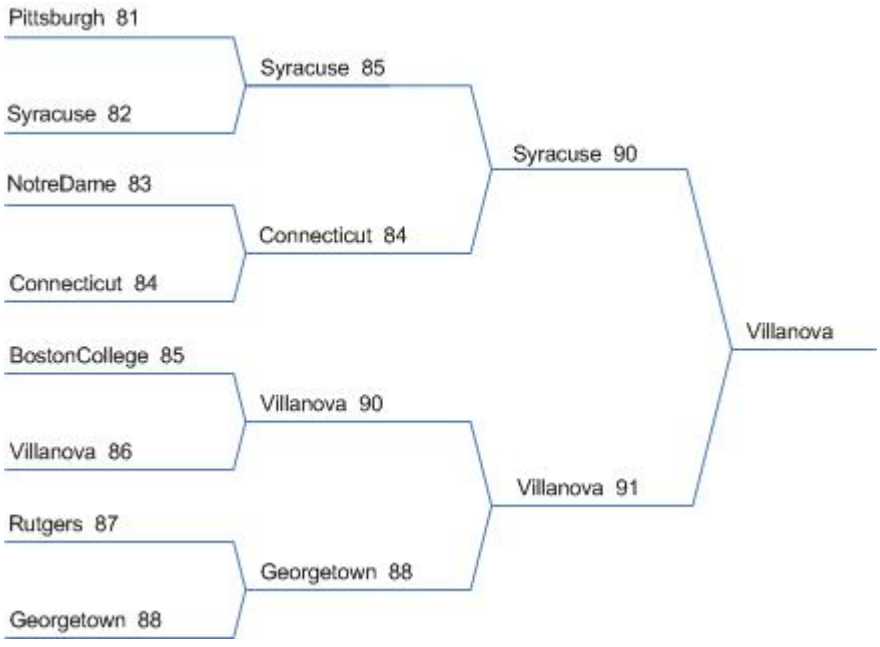
JAVA programmers: your program name must be Prob14.class

Task (problem) Description:

The NCAA hoops tournament is just around the corner, and soon, everyone will be filling out their brackets. Before you make your predictions, you decide a little research is in order. As part of your research, you would like to look at last year's tournament to see how various teams fared. Unfortunately, the only data you can find is a simple listing of the scores for each game of the tournament. Even worse, they are in random order!

To make it easier on yourself, you decide to write a program that will read in all the scores, then print out an easy to read report that lists each game and score by round. While there are always 64 teams in the NCAA tournament (excluding the "play-in" game loser), your programs should work for any 2^n (n>0) number of teams.

Consider the following bracket as you look at the example.



Program Input:

The input file (c:\codewars\prob14.in) consists of the results of each game of the tournament. Each game will be shown as two lines of input. The first line begins with the team listed on the top line of the bracket followed by a space and their score. The second line contains the bottom team. A blank line separates all the games. Team names will not contain any spaces and be no longer than 25 characters.

```
Syracuse 85
Connecticut 84

Pittsburgh 81
Syracuse 82

Rutgers 87
Georgetown 88

BostonCollege 85
Villanova 86

Syracuse 90
Villanova 91

NotreDame 83
Connecticut 84

Villanova 90
Georgetown 88
```

Program Output:

You will output a report to a file (prob14.out). Your report will first show the winner of the tournament. Then, beginning with the last round and working toward the first, print out the round number followed by the results of each game in that round. Each game should be listed in the order (top to bottom) they would appear on a tournament bracket. You must follow the same punctuation and spacing as in the output below.

```
Winner:
Villanova

Round 3:
Syracuse 90
Villanova 91

Round 2:
Syracuse 85
Connecticut 84

Villanova 90
Georgetown 88

Round 1:
Pittsburgh 81
Syracuse 82

NotreDame 83
Connecticut 84

BostonCollege 85
Villanova 86

Rutgers 87
Georgetown 88
```

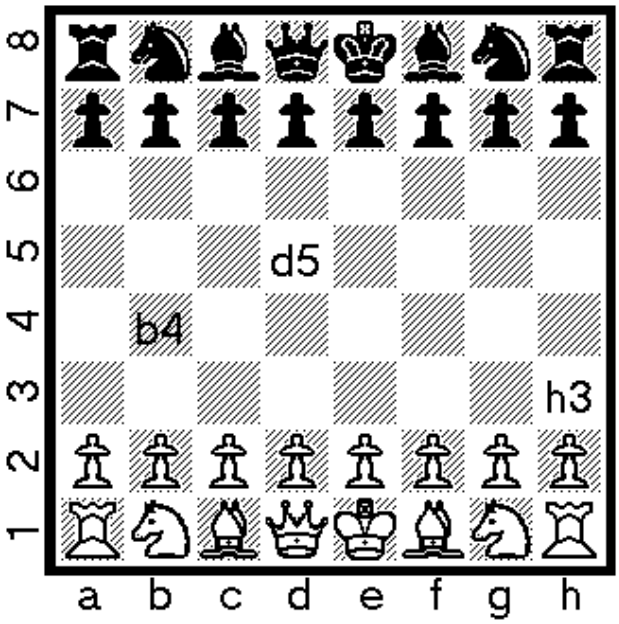
Chess Moves

Problem #15 **Advanced** **20 Points**

C programmers: your program name must be: prob15.exe
JAVA programmers: your program name must be prob15.class

Task (problem) Description:

Chess notation is a shorthand method of replaying a chess match move by move. To understand chess notation, consider an 8x8 chess board.



The rows are denoted by numbers 1-8 and the columns are denoted by letters a-h. To find the coordinates of a certain square, just look at the column letter and row number.

Each piece has a letter associated with it except the pawn. Its starting position(s) is also noted:
K = King (White=e1;Black=e8)
Q = Queen (White=d1;Black=d8)
B = bishop (White=c1,f1;Black=c8,f8)
N = knight(White=b1,g1;Black=b8,g8)
R = rook (White=a1,g1;Black=a8,g8)

White's pawns are lined up on row 2 and black's on row 7.

Every time you move, you write the letter for the piece that moved followed by the name of the square it moved to. For example, Ke2 or Qh5 or Bh8 or Nf6. No dashes; it's that easy. For pawn moves you don't

write the “p”, just the destination square: e4, e5, h8. Remember: Capital letters are pieces, small letters files

When you capture with a piece, put an “x” between the name of the piece and capture square: Kxe2, Qxh5. When you capture something with a pawn, instead of putting the name of the piece on the left, you put the column (not the square) it came from, then the x and the square on which it captured. For example: dxe5 or hxg6.

Captures are NEVER implicit; they will always be denoted with an ‘x’.

When a pawn reaches the final rank it can be promoted to any other chess piece. Indicate the square and the promoted piece as follows: a8Q, c1N. Thus, the pawn at a8 is now a Queen, and the pawn at c1 becomes a knight.

There are other chess moves such as castling and capturing en-passant, but for the sake of this exercise, we will ignore them. You can also assume that if given Rg2 or Kh5 that only ONE Rook or Knight can legally move to the designated square.

Your program should accept a sequence of moves written in chess notation. At the end of the input your program will print an ASCII representation of the chess board at the current state of the game. Pieces should be noted by their color and rank. For example, a black knight would be BN and a white pawn is WP. White always moves first.

Program Input:

Your program should read input from a file named prob15.in

```
e4
e5
Nf3
d6
d4
Bg4
dxe5
Bxf3
```

Program Output:

Your program should output the state of the chess board to the screen. There should be exactly three spaces between each piece. Empty squares are donated by ‘xx’ (lowercase).

```
BR   BN   xx   BQ   BK   BB   BN   BR
BP   BP   BP   xx   xx   BP   BP   BP
xx   xx   xx   BP   xx   xx   xx   xx
xx   xx   xx   xx   WP   xx   xx   xx
xx   xx   xx   xx   WP   xx   xx   xx
xx   xx   xx   xx   xx   BB   xx   xx
WP   WP   WP   xx   xx   WP   WP   WP
WR   WN   WB   WQ   WK   WB   xx   WR
```