Code Warrior Introduction

Problem #1
Novice
2 points

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
```

Sales Tax

Problem #2
Novice
3 points

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

Task Description

Develop a program that calculates the sales tax of an item purchased at one of the local stores.

Program Input

As input, the user will specify the tax rate for the municipality (i.e. 8.25%) and cost of the item. You can assume the user will enter a number between 0 and 1000 that will include no more than 2 decimals places. The input will not include the % or the $ symbols.

Program Output

Output the sales tax (rounded to the nearest hundredth) and the total cost of the item (cost of the item + sales tax).

Sample Program Input / Output

```
Enter Sales Tax: 8.25
Enter Cost of item: 100.00

Your sales tax is: $8.25
The total cost of the item (including sales tax) is: $108.25.
```

Backyard Pond Sizer

Problem #3
Novice / Advanced
4 points

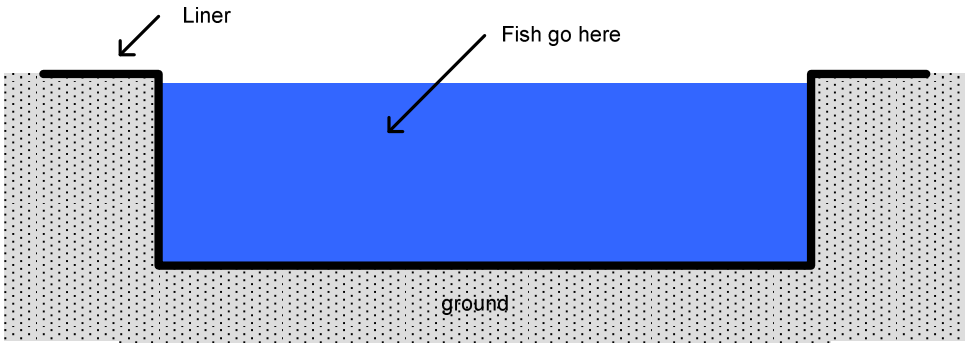
C programmers: your program name must be: prob03.exe
JAVA programmers: your program name must be: Prob03.class

Task Description

Your home and garden shop is growing rapidly, and you've added backyard pond supplies to your store. The customers know their pond's dimensions, but you need a different set of measurements. To properly size the pump, you need to know how many gallons of water the pond will hold. The number of fish the pond can maintain is dictated by the pond's surface area. And you need to calculate the size of the rubber sheet (called a "liner") that creates the bottom and sides of the pond.

Being an excellent computer programmer, you decide to write a "pond sizer" that will calculate these values, given the pond's dimensions. Your program must do the following:

- Calculate the surface area (in square feet) of any size circular or rectangular pond
- Calculate the number of gallons the pond can hold (a cubic foot of water is equal to 7.5 gallons)
- Compute the size of the rubber liner necessary to line the bottom and sides of the pond. The liner must also leave an extra foot of material along all outside edges of the pond.



Note that liner material is purchased in one rectangular piece; for circular ponds, extra material will be trimmed off. The liner size should be expressed in terms of length and width (in feet) rounded up to the next whole foot.

Area of Rectangle = Length × Width Area of Circle = $\pi \times (\text{diameter} / 2)^2$ Volume = Area × Depth

Program Input

Prompt for the type of pond to be constructed – either circular (enter a "C") or rectangular (enter an "R"). Then, prompt for the pond's dimensions. For circular ponds, prompt for the diameter (in feet) and the depth (in inches). For rectangular ponds, prompt for length, width (in feet), and depth (in inches).

Program Output

Your program will output to the screen the surface area of the pond (in square feet, round down), the amount of water the pond will hold (in gallons, round down), and the size of the liner necessary (length and width, in feet, rounded up).

Sample Program Input / Output

```
Enter type of pond (Circular or Rectangular):  R
Enter the length of the pond (in feet):  3
Enter the width of the pond (in feet):  4
Enter the depth of the pond (in inches):  18
```

```
Pond has a surface area of 12 square feet.
Pond will hold 135 gallons of water.
Required liner size is: 8 by 9 feet.

Enter type of pond (Circular or Rectangular):  C
Enter the diameter of the pond (in feet):  5
Enter the depth of the pond (in inches):  24

Pond has a surface area of 19 square feet.
Pond will hold 294 gallons of water.
Required liner size is: 11 by 11 feet liner.

Enter type of pond (Circular or Rectangular):  R
Enter the length of the pond (in feet):  12
Enter the width of the pond (in feet):  33
Enter the depth of the pond (in inches):  44

Pond has a surface area of 396 square feet.
Pond will hold 10890 gallons of water.
Required liner size is: 22 by 43 feet.
```

Benford's Law

Problem #4
Novice / Advanced
4 points

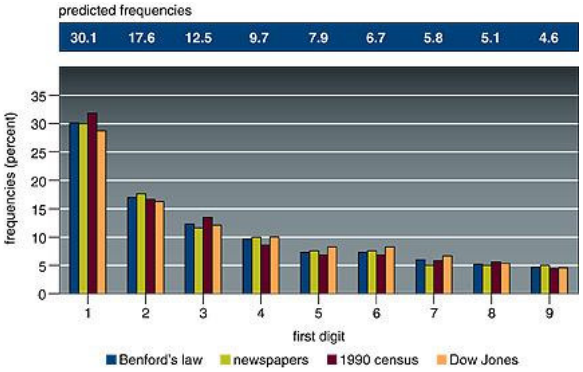
C programmers: your program name must be: prob04.exe
JAVA programmers: your program name must be: Prob04.class

Task Description

Benford's Law is named after the late Dr. Frank Benford, a physicist. In analyzing various kinds of number sets (everything from baseball statistics to street addresses), Dr. Benford noticed that the probability of the first digit of any number being 1 was around 30%, significantly higher than any other number. Today, Benford's Law is used as a powerful and relatively simple tool for pointing suspicion at frauds, embezzlers, tax evaders and sloppy accountants. This is because these people tend to make up their numbers at random.

Your task is to read a series of numbers from the input file, separated by spaces, and determine the frequencies of the numerals 1 – 9 as the **first digit** of a number (there will be no leading zeroes). You will then print to the screen a summary of the occurrence frequencies of each of the numerals. Frequencies should be rounded down to the nearest whole number.

Frequency can be determined by dividing the number of instances found by the total population sampled.



Program Input

Sample data is contained in Prob04.in. Read this data using one of the methods described on your "Contest Instructions" sheet, under "Program Input/Output". Your program will read a set of integer numbers. There will be one number per line. The set of numbers is terminated by the number 0.

```
14
32934
28723470
2727
.
.
.
2787
118
0
```

Program Output

```
Benford's Law Frequencies
Population includes 726 numbers.
Numeral  Sightings  Freq.
   1        208      28%
   2        130      17%
   3         87      11%
   4         73      10%
   5         68       9%
   6         44       6%
   7         36       4%
   8         51       7%
   9         29       3%
```

Application Overload

Problem #5
Novice / Advanced
5 points

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

Task Description

Can I run all of these applications on one HP ProLiant Server? Based on application requirements, determine whether a given set of applications can be simultaneously executed on a single HP ProLiant Server.

Program Input

Sample data is contained in Prob05.in. Read this data using one of the methods described on your "Contest Instructions" sheet, under "Program Input/Output". Determine whether or not the given hardware platform can support the target workload. The first line of input contains server specifications including model number, CPU speed (in MHz) and available memory (in MB) (separated by commas). The next line contains the number of applications. The rest of the lines contain the data for each application. The data for each application contains the application name, CPU and memory requirements for the work targeted to run on the server (separated by commas).

```
HP ProLiant DL385,3600,1600
3
Microsoft IIS,250,500
Payroll,2000,750
Microsoft Exchange,1000,350
```

Program Output

Calculate the total CPU and memory requirements for the entire workload and compare the results to the server specifications which were included in the information file. Your program should determine whether the workload can be executed on the given server and output a yes/no answer. Additionally, calculate and display the percentage of CPU and memory to be used compared to the total available. Percentages should be rounded to 2 decimal places.

```
Yes
90.28% CPU
100.00% memory
```

Roman Numeral Calculator

Problem #6
Novice / Advanced
7 points

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

Task Description

In this problem you are going to create a Roman numeral calculator. This calculator will need to accept two numbers (in Roman numeral form) and then allow a single mathematical operation to be performed. Some examples of equations your calculator should be able to handle are:

XI + V = XVI
X – V = V
X / V = II
II * V = X

The Roman numerals you will need to comprehend are:

I = 1 V = 5 X = 10 L = 50 C = 100 D = 500 M = 1000

To represent other values, these symbols, and multiples where necessary, are concatenated, with the smaller-valued symbols written further to the right. For example, the number 3 is represented as ``III", and the value 73 is represented as ``LXXIII". The exceptions to this rule occur for numbers having units values of 4 or 9, for tens values of 40 or 90, and for hundreds values of 400 and 900. For these cases, the Roman numeral representations are ``IV" (4), ``IX" (9), ``XL" (40), ``XC" (90), “CD” (400), and “CM” (900). So the Roman numeral representations for 24, 39, 49, 449 and 999 are ``XXIV", ``XXXIX", ``XLIX", “CDXLIX” and ``CMXCIX", respectively.

Assume each number will be less than 4,000, and that the results of any requested division operation will be whole numbers. Prompt for the first operand, the operator, the second operand and then display the results.

Operand1 Operator Operand2 = Answer

Program Input

```
Input Operand 1:  XXV
Input Operator: -
Input Operand 2 : V
```

Program Output

```
The result is: XX
```

Sudoku Judging

Problem #7

Novice / Advanced

8 points

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

Task Description

The numbers game of Sudoku has become popular around the world. You've been asked to judge a Sudoku contest, so you decide to write a program to do the judging for you. The game itself is deceptively simple – given a 9x9 grid with some spaces already populated with numbers, you place the digits 1-9 into the remaining spaces. There are only two rules:

- Each row and column must contain each of the digits 1-9 exactly once.
- Each 3x3 region of the grid must contain each of the digits 1 through 9 exactly once.

	5	7				9		
	8		1	5		6		
	6			2		3		5
			8		2			4
1			9		6			2
3			5		4			
5		4		9			8	
		2		6	1		9	
		9				4	7	

Sudoku Solution

2	5	7	6	4	3	9	1	8
4	8	3	1	5	9	6	2	7
9	6	1	7	2	8	3	4	5
7	9	6	8	3	2	1	5	4
1	4	5	9	7	6	8	3	2
3	2	8	5	1	4	7	6	9
5	1	4	3	9	7	2	8	6
8	7	2	4	6	1	5	9	3
6	3	9	2	8	5	4	7	1

Program Input

Sample data is contained in Prob07.in. Read this data using one of the methods described on your "Contest Instructions" sheet, under "Program Input/Output". Each line of the grid will contain the 9 digits with no spaces.

Program Output

Your program must output to the screen the judging result of each submission: CORRECT if the solution is valid, and INCORRECT if it is not.

Sample Program Input / Output

```
257643918
483159627
961728345
796832154
145976832
328514769
514397286
872461593
639285471

CORRECT
```

Simply Sets

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

A set is a collection of objects considered as a whole. The objects of a set are called elements or members. The elements of a set can be anything: numbers, people, letters of the alphabet, other sets, and so on. Sets are conventionally denoted with capital letters, A, B, C, etc

There are several ways to construct new sets from existing ones:

- Two sets (A and B) can be "added" together. The new set is said to be the **Union** of A and B.

  A = { 1, 2, 5, 7}
  B = { 1, 2, 3, 6}

  A + B = { 1, 2, 3, 5, 6, 7 }

- A new set can also be constructed by determining which members two sets (A and B) have "in common". The new set is said to be the **Intersection** of A and B. If no elements are contained in both sets, then A and B are said to be **Disjoint**.

  A = { 1, 2, 5, 7}
  B = { 1, 2, 3, 6}

  A ∩ B = { 1, 2 }

- Two sets (A and B) can also be "subtracted". The **Relative Complement** of A in B (B-A) is the members of set B that are not in set A.

  A = { 1, 2, 5, 7}
  B = { 1, 2, 3, 6}

  B – A = { 3, 6 }

A set may have zero members. This set is referred to as an empty set of a **Null** set.

Your task will be to construct a new sets out of two given set and one of the set operations explained above.

Program Input

Sample data is contained in Prob08.in. Read this data using one of the methods described on your "Contest Instructions" sheet, under "Program Input/Output". Your program should read multiple sets of three-line instructions. The first line will represent the elements in Set A and the second, Set B. An element may be a number, letter, or any alpha-numeric string. Each element is separated by a comma. The third line will be one of the following three operations: **Union, Intersection,** or **Complement**. This series may repeat itself indefinitely, but will always contain two sets followed by an operation.

Program Output

You are to perform the operation given on the previous two sets and output the new set. Each new set should appear on a new line, and each element of the set should be separated by a common. No extra

whitespace is permitted. For empty sets, your program should output the word, **Null**. For the **Complement** operation, you should output the relative complement of A in B (B – A). Your output set should list all of the elements (in order presented) that come from set A followed by the elements (in order presented) from set B. In other words, there is exactly one way to order the elements in your output set.

Sample Program Input / Output

Sample Input:

```
1,Green,red,4,6
1,2,Red,8,3
Union
1,5,7,D,B
D,5,7,Green
Intersection
1,10,55
2,6,8,9
Intersetion
Blue,33,9,D
Blue,67,8,D
Complement
```

Sample Output:

```
1,Green,red,4,6,2,Red,8,3
5,7,D
Null
67,8
```

Random Sort

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

Since the dawn of computing, the sorting problem has attracted a great deal of research, perhaps due to the complexity of solving it efficiently despite its simple, familiar statement. Efficient sorting is important to optimizing the use of other algorithms (such as search and merge algorithms) that require sorted lists to work correctly.

Your task is simple. Alphabetize and output a given list of words. Seems straightforward, but where would be the fun in that? To mix things up a bit—pun intended—you will be given a new alphabet along with the wordlist in which to work. Happy sorting!

Program Input

Sample data is contained in Prob09.in. Read this data using one of the methods described on your "Contest Instructions" sheet, under "Program Input/Output". The first line of input will be your alphabet which will be a string of upper case letters. The rest of the input will be a list of words—one word per line—to be sorted. Words will consist of both upper and lower case letters. There are no bounds to the number of words. Your program should detect the end of input.

Program Output

Your program should output the alphabetized list of words--one word per line.

Sample Program Input / Output

Sample Input:

```
QWERTYUIOPASDFGHJKLZXCVBNM
hat
cat
bat
book
bookworm
Dallas
Austin
Houston
fire
firefox
fumble
```

Sample Output:

```
Austin
Dallas
fumble
fire
firefox
Houston
hat
cat
book
bookworm
bat
```

DVR Scheduling

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

The digital video recorder (DVR) is becoming a popular item for home entertainment systems. One of the benefits of a DVR is that it can keep track of when your favorite shows are on for you. Simply tell the DVR which shows you like to watch, and it will record them for you.

Unfortunately, many of your favorite shows are on at the same time and your generic DVR can only record a single show at once. Additionally, the DVR will randomly choose which show to record when there is a scheduling conflict. There must be a better way to do this.

Your task it is to write the conflict resolution code for a new Open Source DVR program. The new DVR should consider two things. First, whether another airing of the same show is available to be recorded. Second, if all airings of a particular show conflict with another show(s), then the show(s) with the highest priority will be recorded. In other words, you may have to consider the priority of more than two shows when resolving conflicts. You should only record a show once and during its first available airing.

Develop a program that takes as input the taping schedule of a DVR and determines which shows are recorded and at what time.

Program Input

Sample data is contained in Prob10.in. Read this data using one of the methods described on your "Contest Instructions" sheet, under "Program Input/Output". The input data contains a list of shows to record and the dates and times they will be aired. The shows are listed in priority order, with the first show having the highest priority. Each line contains the name of the show enclosed by quotes (") followed by a series of days and times.

Each day and time consists of a three-letter abbreviation for a day of the week (i.e., one of SUN, MON, TUE, WED, THU, FRI, or SAT), followed by a space, followed by the time of day, expressed in ``military time'', i.e., as a four-digit sequence whose first two digits are in the range 00..23 (indicating the number of hours since midnight) and whose last two digits are in the range 00..59 (indicating the number of minutes since the beginning of the current hour).

All shows start on the hour and will last less than one hour.
The airings for each show are listed in no particular order.
There is no limit to the number of airings a show may have.

Program Output

For each show listed in the input (and in the same order), display the show name and the time it will be recorded that week. If the show can not be scheduled, you should output "Will not be recorded."

Sample Program Input / Output

Sample input :

```
"Veronica Mars" WED 2000 SUN 2200
"Lost" WED 2000
"Smallville" THU 1900
"The O.C." THU 2000
"24" MON 2000
"Scrubs" TUE 2000
"American Idol" TUE 2000 WED 2000
```

"The Family Guy" SUN 2000
"The Simpsons" SUN 1900
"Desperate Housewives" SUN 2000 TUE 1900

Sample Output :

Veronica Mars SUN 2200
Lost WED 2000
Smallville THU 1900
The O.C. THU 2000
24 MON 2000
Scrubs TUE 2000
American Idol Will not be recorded
The Family Guy SUN 2000
The Simpsons SUN 1900
Desperate Housewives TUE 1900

How Many Ways to Make Change?

Problem #11
Novice / Advanced
12 points

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

Task Description

Did you know there are 292 different ways to make change for a dollar? Yes, using a combination of half-dollars, quarters, dimes, nickels, and pennies, there are really that many!

Your task is to write a program to determine how many different ways there are to make change for any amount between $0.01 and $10.00. Your solution must allow for the use of half-dollars ($0.50), quarters ($.25), dimes ($.10), nickels ($.05) and pennies ($.01). Note that the order of the coins used does not matter (two dimes and a nickel is the same way as a nickel and two dimes), and therefore each combination is only counted once.

Program Input

Your program must prompt for the amount of money to count. For simplicity, the amount will be entered as the number of cents to count, between 1 and 1000 (which equates to $0.01 to $10.00).

Program Output

Your program must output to the screen the total number of coin combinations that can produce the desired amount.

Sample Program Input / Output

```
Enter amount of change to make (in cents):  100

    There are 292 ways to make change for $1.00

Enter amount of change to make (in cents):  10

    There are 4 ways to make change for $0.10

Enter amount of change to make (in cents):  4

    There are 1 ways to make change for $0.04
```

Mayan Calendar

Problem #12
Novice / Advanced
15 points

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

Task Description

The calendar of the Maya is different from our calendar. The Mayan "Long Count" is a calendar system that consists of 5 numeric values written as follows: 12.19.1.17.5. Moving from right to left, the definitions of each component of the Mayan Long Count follow:

- The first (rightmost) is a kin, which represents a single day.
- The second is a uinal. There are 20 kin in 1 uinal.
- The third is a tun. There are 18 uinal in 1 tun.
- The fourth is a katun. There are 20 tun in 1 katun.
- The fifth (leftmost) is a baktun. There are 20 katun in 1 baktun.

Thus, 12.19.1.17.5 is read, "12 baktun, 19 katun, 1 tun, 17 uinal, 5 kin.

Given that 12/23/2012 is the date 13.0.0.0.0, write a program that reads a "normal" calendar date from standard input and outputs that date to standard output in Mayan Long Count format. Your program will not have to handle any data inputs that might result in a negative Long Count. Thus, the minimum output date your program might produce is 0.0.0.0.0. Finally, the input date will be between 01/01/0001 and 12/23/2012, inclusive. Input date formats will always be mm/dd/yyyy, padded with zeroes wherever necessary.

Hint #1: The maximum value for each component is equal to the number of units in its collection minus 1. Thus, there will never be more than 17 uinal or 19 tun, for example. One more of either turns the number over to the next highest component.

Hint #2: To calculate a leap year in our calendar, do the following: every year divisible by 4 contains an extra day (Feb. 29) except a century year that is not evenly divisible by 400. Thus, the years 1700, 1800, 1900 and 2100 are not leap years, but 1600, 2000, and 2400 are. Other leap years include 1200 and 800 but not 1400 or 1300, and so on.

Program Input

```
12/22/2012
```

Program Output

```
12.19.19.17.19
```

Tumbled Tiles

Problem #13
Novice / Advanced
18 points

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

Task Description

A popular puzzle consists of nine square tiles which must be arranged in a 3x3 grid. On each edge of each tile is printed the Head or Feet of a frog. Frogs may be Red, Yellow, Green, or Blue. Each of the nine puzzle tiles is distinct. The tiles must be rotated and arranged such that the edges of any two connected tiles form a color-matched pair of head to feet (Fig. 1). For any given tile set, there may be more than one solution.

Program Input

Sample data is contained in Prob13.in. Read this data using one of the methods described on your "Contest Instructions" sheet, under "Program Input/Output". The data file consists of nine lines, where each line represents a puzzle tile. Each tile has a number, followed by four edges, given clockwise: top, right, bottom, left. Each edge is described by a color and body part.

```
1,YH,BH,RF,GH
2,GH,YF,RH,BH
3,YH,RF,BF,GF
4,GF,YF,RF,GH
5,GF,RF,YF,BH
6,YH,BH,YH,BF
7,GF,RH,RF,BH
8,GH,YF,BF,RH
9,YF,GH,RF,BH
```

Fig. 1. Two matching tiles.

Program Output

The program will write a puzzle solution to the screen. Each edge of each puzzle tile must match the edges of each adjacent puzzle tile. The program will also print each tile number in the center of the tile. Please use the following format:

```
   YF        BH        RF
GH- 2-RH RF- 7-GF GH- 9-BH
   BH        RH        YF
   BF        RF        YH
YH- 6-YH YF- 4-GH GF- 3-RF
   BH        GF        BF
   BF        GH        BH
YF- 8-RH RF- 1-YH YF- 5-GF
   GH        BH        RF
```