UW/TEALS Programming Competition
December 15th, 2012

ADVANCED DIVISION

- Do the problems in any order you like.
- 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**
1. Best Number…		
2. Factorials		
3. Grocery Trip		
4. Above Average		
5. CD Collection		
6. Crazy Conversions		
7. Day To Day		
8. Flex J Box		
9. Race		
10. Forests		
11. Attendance		
12. Taxi, Taxi!		
13. Test Parity		
14. Weird Change		
15. Driving to Alaska		

Good luck!

1. Best Number…

Input File: bestnumber.dat

General Statement: You live in a world where you only know how to multiply and divide, and an evil arithmetic wizard has charged you with the task of finding the "best" number. Trouble is, sometimes he changes his mind on what the best number is! The best number is always either the biggest number or the smallest number that you can make.

Input: The first line in the input file is an integer, *n*, representing the number of integers in the data set. The second line always contains the word "biggest" or "smallest". The *n* following lines each have one real number.

Output: The biggest or smallest number that you can make by multiplying or dividing the numbers in the list.

Assumptions – Helpful Hints: You may assume that the second line contains either "biggest" or "smallest" followed by a newline character. You may assume that the list of numbers that follows is at least 1 row long. There will only be one number per row and all numbers will decimals greater than 0. You are only allowed to use the "*" and the "/" operators. Both operators may be used together to calculate the "best" number. In the example below, the best number is `1.7 * 3.2 / 0.5 = 10.88` because that is the biggest number that can be generated by using only the 3 input values and the two operators. You may assume that the best number will never be so large or so small as to cause overflow errors.

Example Input File:
```
3
biggest
1.7
3.2
.5
```

Example Output to screen:
```
10.88
```

2. Factorials

Input File: factorial.dat

General Statement: A factorial, denoted by n!, is the product of all positive integers from 1 to n. For example, 10! = 10*9*8*7*6*5*4*3*2*1 and 6! = 6*5*4*3*2*1.

Input: An integer N.

Output: N!

Assumptions – Helpful Hints: You may assume that 13>n>0.

Example Input File:
```
6
```

Example Output to screen:
```
720
```

3. Grocery Trip

Input File: GrocerySample.dat

General Statement: Suman and Rajesh need to go to the store to get tomatoes. Unfortunately, it's too far to walk, but (fortunately) they each have a large menagerie (collection of animals) which they can ride there. They want to show up in style, though, which means that they must show up on matching animals. You will write a program that decides whether this is possible.

Input: The first line of the input is a number n. The next n lines each contain one animal in Suman's collection (e.g., Horse, Pig, Donkey, etc.). The next line is a number m, and the following m lines each contain one element of Rajesh's collection.

Output: If there is an animal that Suman and Rajesh both have in their collections, the output should be "Tomatoes for youza!" (formatted exactly like that: spaces between words, 'T' capitalized). Otherwise, the output should be "No tomatoes for youza!".

Assumptions – Helpful Hints: You may assume that m and n are both less than 25, that both are nonzero, and that if there is a matching animal then it is spelled exactly the same (including the same capitalization) on both lists.

Example Input File:
```
5
Elephant
Cow
Water Buffalo
Squirrel
Pig
3
Hippo
Deer
Water Buffalo
```

Example Output to screen:
```
Tomatoes for youza!
```

4. Above Average

Input File: average.dat

General Statement : It is said that 90% of college freshmen expect to be above average in their class. You are to provide a reality check.

Input: The first line of the input contains an integer C, the number of test cases. C data sets follow. Each data set begins with an integer, N, the number of people in the class (1 <= N <= 1000). N integers follow, separated by spaces or newlines, each giving the final grade (an integer between 0 and 100) of a student in the class.

Output: For each case, you are to output a line giving the percentage of students whose grade is above average, rounded to 3 decimal places.

Example Input File:
```
5
5 50 50 70 80 100
7 100 95 90 80 70 60 50
3 70 90 80
3 70 90 81
9 100 99 98 97 96 95 94 93 91
```

Example Output to screen:
```
40.000%
57.143%
33.333%
66.667%
55.556%
```

5. CD Collection

Input File: cds.dat

General Statement : Jack and Jill have decided to sell some of their Compact Discs, while they still have some value. They have decided to sell one of each of the CD titles that they both own. How many CDs can Jack and Jill sell? Neither Jack nor Jill owns more than one copy of each CD.

Input: The input consists of a sequence of test cases. The first line of each test case contains two non-negative integers N and M, each at most one hundred, specifying the number of CDs owned by Jack and by Jill, respectively. This line is followed by N lines listing the catalog numbers of the CDs owned by Jack in increasing order, and M more lines listing the catalog numbers of the CDs owned by Jill in increasing order. Each catalog number is a positive integer no greater than one billion. The input is terminated by a line containing two zeros. This last line is not a test case and should not be processed.

Output: For each test case, output a line containing one integer, the number of CDs that Jack and Jill both own.

Example Input File:
```
3 3
1
2
3
1
2
4
0 0
```

Example Output to screen:
```
2
```

6. Crazy Conversions

Input File: conversions.dat

General Statement: Flash, the mathematician, needs your help with some tricky formulas. In a momentary loss of computing ability, he needs you to compute the following strange formulas into their final value. In his madness, he chose to name his formulas after his pets, Crash, Dash, Mash, and Trash.

Crash = $\frac{A}{4} + 7B$

Dash = $\frac{A+B^2}{A+\frac{1}{B}} + \frac{A}{B}$

Mash = $\frac{1}{\frac{1}{A}-\frac{1}{B}}$

Trash = $\frac{4}{\frac{A}{B}}\left[\frac{1+\frac{5}{C+D}}{A}\right]^{\frac{1}{2}} - \frac{A}{C+D}$

Input: The first line in the file will contain an integer representing the number of data sets to follow. Each data set will contain 4 integers representing A, B, C, and D (in that order).

Output: The values of the expressions Crash, Dash, Mash, and Trash (in that order). An error tolerance of +-0.01 will be accepted. One blank line should separate each set of output.

Example Input File

```
2
12 4 3 1
8 4 9 3
```

Example Output to screen:

```
31.00
5.29
-6.00
-2.96

30.00
4.91
-8.00
-0.61
```

7. Day To Day

Input File: daytoday.dat

General Statement : Given two dates, calculate and output the number of days between the two dates (exclusive). For example, there are 19 days between June 2, 2012 and June 22, 2012.

Input: The first line in the file will contain an integer representing the number of data sets to follow. Each data set will consist of six integers representing two dates. Each date will be represented by a month, followed by a day, followed by a year. The second date is guaranteed to be after the first date by at least one day, but could be in a later year. Your program has to deal with dates ranging from 1/1/1901 to 12/31/2099. Recall that leap years have an extra day (February 29) and that years divisible by 4 are leap years.

Output: The number of days between each pair of dates (exclusive).

Example Input File
```
3
6 2 2012 6 22 2012
7 4 2011 12 25 2011
12 28 1980 1 1 1981
```

Example Output to screen:
```
19
173
3
```

8. Flex J Box

Input File: flexjbox.dat

General Statement: Given four positive integers output a box made of asterisks with a **J** at the given position.

Input: The first line in the file will contain an integer representing the number of data sets to follow. Each data set will contain four integers. The first two integers represent the number of rows and columns (in that order) of the box. The second two integers represent the position of the **J** (row, column). The uppermost left corner's position is (0,0).

Output: The resulting box containing a **J** with one blank line separating each output. The **J** will always be inside of the box, and never outside or on the border.

Example Input File

```
2
4 5 2 2
10 9 3 4
```

Example Output to screen:

```
*****
*****
**J**
*****

*********
*********
*********
****J****
*********
*********
*********
*********
*********
*********
```

9. Race

Input File: gender_race.dat

General Statement: You have been hired to keep track of the winners in a race. Given names, finishing time, and gender for each racer determine the fastest and second fastest time overall, fastest and second fastest among the males, and finally best and second best among the females. There will always be at least two males and two females in the race and never more than 20 racers.

Input: The first line in the file will contain an integer representing the number of racers in the race. Each following line will contain a name, followed by their finishing time, followed by their gender. There will be no ties, no more than 20 racers, and only one race. The name will be a single token with no spaces.

Output: Name, time, and gender for each of the following:

- Overall winner
- Overall second
- Fastest female
- Second fastest female
- Fastest male
- Second fastest male

Example Input File

```
6
Sue 48 F
Kelly 52 M
Stacey 43 F
Kelly 49 F
Stacey 50 M
John 44 M
```

Example Output to screen:

```
Stacey 43 F
John 44 M
Stacey 43 F
Sue 48 F
John 44 M
Stacey 50 M
```

10. Forests

Input File: trees.dat

General Statement: If a tree falls in the forest, and there's nobody there to hear, does it make a sound? This classic conundrum was coined by George Berkeley (1685-1753), the Bishop and influential Irish philosopher whose primary philosophical achievement is the advancement of what has come to be called subjective idealism.

Your task is to compare the opinions of several different people about which trees have fallen and which have not. People may have different opinions as to which trees, according to Berkeley, have made a sound.

Input: A forest contains T trees numbered from 1 to T and P people numbered from 1 to P. The input consists of a line containing T and P followed by several lines, containing a pair of integers i and j, indicating that person i has heard tree j fall. You may assume that P < 100 and T < 100.

Output: The number of different opinions represented in the input. Two people hold the same opinion only if they hear exactly the same set of trees.

Example Input file
```
3 4
1 2
3 3
1 3
2 2
3 2
2 4
```
Output to screen:
```
2
```

11. Attendance

Input File: attendance.dat

General Statement: Schools track tardiness and attendance every day. Your job is to take the data gathered and do a brief analysis on it.

Input: Data gathered for six classrooms. Each line represents the data collected from each class. The first number is the total number of students in the class, the second number is the number of students absent that day, and the third number is the number of tardies that day. Classes will be listed in order from 1 to 6.

Output: The output will be as follows:

1. Total number enrolled in all classes
2. Total number of tardies in all classes
3. The class with the best attendance percentage
4. A list of classes in order from best to worst attendance percentage (if two or more classes have the same attendance percentage, they can be listed in any order)

Example Input File

```
10 1 1
10 0 1
20 3 2
20 6 0
20 1 3
20 4 6
```

Example Output to screen:

```
100
13
2
2 5 1 3 6 4
```

12. Taxi, Taxi!

Input File: taxi.dat

General Statement: The cost to ride a taxi in this land is 50 cents for the first 1/5 mile (or less) and 22 cents for each additional 1/5 mile or part thereof. In addition, the taxi will wait for you while you conduct your business. The cost for waiting is 20 cents per 60 second period or less.

Input: The first line in the file will contain an integer representing the number of taxi rides. Each taxi ride will be represented by two non-negative values separated by a space. The first value is the number of miles traveled, and the second value is the number of seconds of waiting time.

Output: The cost of each taxi ride. Dollar format output required, each answer on one line.

Example Input File:
```
4
1 0
2 0
1.5 75
3.4 125
```

Example Output to screen:
```
$1.38
$2.48
$2.44
$4.62
```

13. Test Parity

Input File: testparity.dat

General Statement: In a somewhat misguided effort to level the testing field, your teacher has decided to assess a tax on all test scores. Test scores will be greater than zero but no higher than 120. Using the following tax table, all test scores will be adjusted so it appears that everyone seems to be doing fairly well, hoping the administration will be pleased.

Score Range:	Action:
0 < score <= 25	Double the score
25 < score <= 50	Increase score by 50%
50 < score <= 75	Increase score by 25%
75 < score <= 100	No change
100 < score <= 120	Decrease score by 10%

Your job is to write a program that will make the appropriate adjustment for each score.

Input: Several test scores, all on one line, each separated by one space.

Output: An integer representing the adjusted scores (rounded to the nearest integer), all on one line, each separated by a space.

Example Input File
```
12 50 80 117
```

Example Output to screen:
```
24 75 80 105
```

14. Weird Change

Input File: weirdChange.dat

General Statement: In this land they do things weird, including their monetary system, where the basic unit is the prog! Each **prog** is worth 3 cents in US currency. The only coins in this system have values of 1, 5, 13, 23, 37 and 47. Your job is to help your programming coach with her money during your visit to this land. The reason she needs help is because of her obsession with giving exact change while using the absolute fewest number of coins necessary for the transaction.

Input: The first line in the file will contain an integer representing the number of **progs** to follow.

Output: The dollar amount equivalent to the given **prog** cost, and six values representing the exact change using the fewest **prog** coins (in descending order of value) needed to make the transaction. Assume your coach always has enough coins to make the purchase.

Example Input File:
```
4
12
41
34
50
```

Example Output to screen:
```
$0.36 0 0 0 0 2 2
$1.23 0 0 1 1 1 0
$1.02 0 0 1 0 2 1
$1.50 0 1 0 1 0 0
```

15. Driving to Alaska

Input File: drive.dat

General Statement: The Alaska Highway runs 1422 miles from Dawson Creek, British Columbia to Delta Junction, Alaska. Brenda would like to be the first person to drive her new electric car the length of the highway. Her car can travel up to 200 miles once charged at a special charging station. There is a charging station in Dawson Creek, where she begins her journey, and also several charging stations along the way. Can Brenda drive her car from Dawson City to Delta Junction and back?

Input: The input contains several scenario. Each scenario begins with a line containing n, a positive number indicating the number of charging stations. n lines follow, each giving the location of a filling station on the highway, including the one in Dawson City (the locations might appear in any order). The location is an integer between 0 and 1422, inclusive, indicating the distance in miles from Dawson Creek. No two filling stations are at the same location. A line containing 0 follows the last scenario.

Output: For each scenario, output a line containing POSSIBLE if Brenda can make the trip. Otherwise, output a line containing the word IMPOSSIBLE.

Example Input file
```
2
0
900
8
1400
1200
1000
800
600
400
200
0
0
```

Output to screen:
```
IMPOSSIBLE
POSSIBLE
```