Computer Science Competition
State 2024
Programming Problem Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Ace
Problem 2	Bart
Problem 3	Ben
Problem 4	Bethany
Problem 5	Conrad
Problem 6	Dilbert
Problem 7	Jason
Problem 8	Lina
Problem 9	Max
Problem 10	Nicholas
Problem 11	Rufus
Problem 12	Victoria

1. Ace

Program Name: Ace.java **Input File: None**

You could call Ace the "Historian of UIL Academics." He has seen it all. He did Number Sense and Slide Rule in high school. He has coached nearly every event. He, to this day, follows the competition by poring through the scores posted for the District, Regional, and State Meets.

If you walk by his house on a Friday afternoon and sit on the porch with him and have some iced tea, he will tell you some stories about the rich history of UIL Academics.

Just the other day, he was telling me about the UIL Computer Science contest. It first came onto the scene in 1990 with a format just a bit different that the format it follows today. When I saw him, he was remembering the 1999 contest. Of course, this year marks the 25th anniversary of that event. That was way back before there was a 6A classification.

He recalls that the 5A champion, Langham Creek, almost missed the Saturday programming portion of the contest. They were dropped off at Painter Hall which had been the site of the Friday night written test. But, programming in 1999 was held in the brand new UIL Building on Manor Road. The team had to hike over a mile uphill with PCs and printers in tow and made it with only minutes to spare. Ah, those were the days.

Let's print out a list of the 1999 UIL CompSci Champions for Ace.

Input: None.

Output: Print the information below that lists the 1999 UIL State CompSci Champions. You will notice there are 6 dashes (minus signs) in the output.

Sample input: None.

Sample output:

```
UIL CompSci State Champion Teams
25 Years Ago - 1999
5A - Houston Langham Creek HS
4A - Fort Worth Dunbar HS
3A - Devine HS
2A - Stockdale HS
1A - Muenster HS
```

2. Bart

Program Name: Bart.java **Input File: bart.dat**

You and Bart have been experimenting with quantum physics lately, and have learned how to make one way portals. Now Bart has devised a test for your portals. He has designed a maze of sorts, with these portals inside, and your job is to navigate through the maze and find the shortest path to escape. He has also coincidentally created a black hole within the maze, which you need to avoid. You will be given a map of the maze, and it will be made up of the following characters:

- `'$'` – This character denotes the starting point of the maze, where you will begin your journey.
- `'^'` – This character denotes the exit from the maze, where you will end your journey.
- `'#'` – This character denotes a wall, an area that is permanently impassable.
- `'.'` – This character denotes an open space, which is passable at a rate of 1 space per second.
- `'[A-Z]'` – These characters denote the entrance to a portal. When you enter a portal, it takes a certain amount of time, then you are at the exit block (the amount of time is equal to the letter's position in the alphabet (a = 1, b = 2, and so on)).
- `'[a-z]'` – These characters denote the exit from a portal. When you enter a portal, it takes a certain amount of time, then you are at the exit block (the amount of time is equal to the letter's position in the alphabet (a = 1, b = 2, and so on)).
- `'@'` – This character denotes the black hole, there will only be one of these per map, and the black hole itself is always impassable. The black hole works as follows:
  - At the beginning of the test, the radius of effect of the black hole is 1.
  - Every 3 seconds, the radius will expand by 1.
  - The radius of the black hole extends in all directions, including diagonals (up, down, left, right, up-right, up-left, down-right, down-left).
  - If you are in the outside 2 blocks (outer radius) of the black hole's radius, your movement speed will be cut in half.
  - If you are inside the radius of the black hole, and not in the outside 2 blocks of the radius, you cannot move (these blocks are impassable.).

You can only move in the 4 cardinal directions (up, down, left, right), and every step takes one second, unless you are in the radius of the black hole. If the spot you are in comes into the radius as you leave it, you are unaffected. For every portal within the maze, there will only be one entrance and exit.

Input: The input will begin with an integer, `n` (`0 < n <= 1000`), denoting the number of test cases to follow. Each test case will begin with 3 space-separated integers, `r` and `c`, denoting the number of rows and columns in the maze. Each of the following `r` lines will each contain `c` character denoting the maze.

Output: If it is possible to reach the exit, output the string `"Exit is short for exciting."`, followed by a space and the number of steps it took to reach it. If it is not possible to reach the exit, output `"Tell Matthew Mcconaughey I said hi."`.

~ ***Sample input and output on next page*** ~

~ ***Bart continued*** ~

Sample input:

```
2
7 7
$..#.A#
#....##
##....#
@...###
#D....a
###..##
d^...##
4 8
#$##....
J....##B
########
^.j.b..@
```

Sample output:

```
Tell Matthew Mcconaughey I said hi.
Exit is short for exciting. 15
```

3. Ben

Program Name: Ben.java **Input File: ben.dat**

Ben was recently creating his account for a new game that he wanted to play with his friends and went to enter in his username. After successfully creating his account and joining his friends' lobby, one of his friends noticed that he spelled his username incorrectly! Ben had accidentally spelled "pigeon" as "pidgeon" as he thought that "pidgeon" sounded more phonetically correct than "pigeon."

Being the good friends that they are, Ben's friends decided to poke a little fun at him by changing their usernames to other misspellings of pigeon. More specifically, they wanted to create usernames which contain multiple noncontiguous substrings of the string "pidgeon". Help Ben determine just how good of friends he has by calculating the number of unique noncontiguous substrings there exist of the string "pigeon" in the various different usernames his friends came up with. Given that these numbers may become decently large, determine this number modulo $10^9 + 7$.

Input: The first line of input will consist of a single integer n ($1 \leq n \leq 50$) denoting the number of testcases to follow. The next n testcases will consist of a single line which consists of the username (u) for the current testcase. It is guaranteed that u consists of only the letters 'p', 'i', 'd', 'g', 'e', 'o', and 'n', although, not necessarily in that order. Additionally, it should be noted that the length of the string will fall in the range $1 \leq |u| \leq 10^3$.

Output: For each of Ben's n usernames, print out the number of non-contiguous substrings of the username that are equivalent to the string "pidgeon" modulo $10^9 + 7$.

Sample input:

```
4
pidgeon
pidgeonpidgeon
pigeon
ppiiddggeeoonnppiiddggeeoonn
```

Sample output:

```
1
8
0
1024
```

4. Bethany

Program Name: Bethany.java **Input File: bethany.dat**

Bethany loves numbers. She is a Number Sense expert. As part of her study, she has memorized the number of factors that each integer from 1 to 100 has. She has also learned various techniques for quickly determining the number of factors for any number.

She wants to see what is in common for integers that have the same number of factors

Write a program that inputs two numbers, N and F. Print the first N integers which have exactly F factors.

Example: If the inputs were 5 and 4, she would want to see the first 5 integers which all have 4 factors. Those numbers would be 6, 8, 10, 14, and 15.

Input: The first line of data will be an integer T in the range [1,20]. T represents the number of data lines to follow. Each line of data will contain two integers N and F. N is an integer in the range [2,20] indicating how many outputs should be created for that data set. F is an integer in the range [2,100]. F represents the factor count for each output for that data set. One space shall separate N and F.

Output: Each test case will produce 1 line of output containing at least 2, but not more than 20 positive integers. Each line will print the first N positive integers which have exactly F factors. There will be one space between each number in each line of outputs.

Sample input:

```
5
10 2
5 6
8 4
2 9
10 8
```

Sample output:

```
2 3 5 7 11 13 17 19 23 29
12 18 20 28 32
6 8 10 14 15 21 22 26
36 100
24 30 40 42 54 56 66 70 78 88
```

5. Conrad

Program Name: Conrad.java **Input File: conrad.dat**

Conrad is learning about words that have CCPD, a "constant common positive difference."

To have a constant common positive difference, a word must have at least two letters. Then, moving left to right through the word one should consider each pair of consecutive letters. If each pair of letters has the same positive difference (considering the position of the letter in the alphabet), that word has a CCPD.

Example #1 - FHJLJH - the positive differences are: 2 (FH), 2 (HJ), 2 (JL), 2 (LJ), and 2 (JH). Each positive difference is 2. So this string has a CCPD.

Example #2 - ABCDCBBC - the positive differences are: 1 (AB), 1 (BC), 1 (CD), 1 (DC), 1 (CB), 0 (BB) and 1 (BC). The differences are not all the same. So this string does not have a CCPD.

Input: The first line of data will be an integer T in the range [1,20]. T represents the number of data lines to follow. Each line of data will contain one string which has a length in the range [2,26]. The string will consist only of uppercase letters.

Output: Each output will either print the original string if it has a constant common positive difference (CCPD), or the message "BAD!!!" written in all uppercase with three exclamation points if it does not have a CCPD..

Sample input:

```
5
GAGA
GULP
FOX
ABCDEFGHIJLMNOPQRSTU
DINS
```

Sample output:

```
GAGA
BAD!!!
FOX
BAD!!!
DINS
```

6. Dilbert

Program Name: Dilbert.java **Input File: dilbert.dat**

Dilbert likes to play around with numbers and has developed an interesting process for manipulating positive integers. He wants to reverse integers and 12345 reversed in decimal form is 54321 which is just too simple. However, the binary form might be more interesting. The binary form of 12345 is **0011 0000 0011 1001** (separated into 4-bit groups for readability) and the reverse of the binary form would be **1001 1100 0000 1100** which has a decimal value of 38848, not 54321! But to make it even more interesting, lets discard leading 0s of the initial binary form before it is reversed. So, the modified binary form now becomes **11 0000 0011 1001** and when reversed produces **10 0111 0000 0011** which has a decimal value of 9987. Hmmm…

Can help Dilbert by writing program to verify his results?

Input: An unknown number of whitespace separated positive integers but no more than 100. Each integer is a single test case and will consist of a single positive integer that will not exceed the value 9×10^{18}.

Output: Each test case will produce 1 line of output containing the decimal equivalent of the reversed binary form followed by a single space and the reversed full binary form without leading 0s

Sample input:

```
10 12345 54321 65536 2903185746
17 1 257 2147483647 1073741984 41943185
```

Sample output:

```
Test Case: 1: 5 101
Test Case: 2: 9987 10011100000011
Test Case: 3: 35883 1000110000101011
Test Case: 4: 1 1
Test Case: 5: 1251528885 1001010100110001101000010110101
Test Case: 6: 17 10001
Test Case: 7: 1 1
Test Case: 8: 257 100000001
Test Case: 9: 2147483647 11111111111111111111111111111111
Test Case: 10: 41943041 10100000000000000000000001
Test Case: 11: 35913733 10001001000000000000000101
```

7. Jason

Program Name: Jason.java **Input File: jason.dat**

Jason's sister was working with 2-dimension arrays for her college math homework. She explained that arrays are called matrices in math and she was performing matrix multiplication. She explained the following example:

Start with two matrices (2-dimension arrays): matrix **A** with **n** rows and **m** columns and matrix **B** with **m** rows and **p** columns. The number of columns **m** in **A** must equal the number of rows **m** in **B**. Their product **C = AB** is a matrix **C** with **n** rows and **p** columns. Position **(i,j)** in the result matrix **C** is the sum of **m** products, the elements in row **i** of **A** multiplied by the elements in column **j** of **B**.

Matrix A n × m 3 × 4	Col 1	Col 2	Col 3	Col 4
Row 1	1	0	-3	2
Row 2	2	-3	0	2
Row 3	-3	0	2	3

Matrix B m × p 4 × 2	Col 1	Col 2
Row 1	-1	0
Row 2	2	-4
Row 3	-3	3
Row 4	0	-2

Matrix C n × p 3 × 2	Col 1	Col 2
Row 1	8	-13
Row 2	-8	8
Row 3	-3	0

C(1,1) sums the products of row 1 from A and col 1 from B
= A(1,1) × B(1,1) + A(1,2) × B(2,1) + A(1,3) × B(3,1) + A(1,4) × B(4,1)
= 1 × -1 + 0 × 2 + -3 × -3 + 2 × 0 = 8
C(1,2) sums the products of row 1 from A and col 2 from B
= A(1,1) × B(1,2) + A(1,2) × B(2,2) + A(1,3) × B(3,2) + A(1,4) × B(4,2)
= 1 × 0 + 0 × -4 + -3 × 3 + 2 × -2 = -13
C(2,1) sums the products of row 2 from A and col 1 from B
= A(2,1) × B(1,1) + A(2,2) × B(2,1) + A(2,3) × B(3,1) + A(2,4) × B(4,1)
= 2 × -1 + -3 × 2 + 0 × -3 + 2 × 0 = -8
C(2,2) sums the products of row 2 from A and col 2 from B
= A(2,1) × B(1,2) + A(2,2) × B(2,2) + A(2,3) × B(3,2) + A(2,4) × B(4,2)
= 2 × 0 + -3 × -4 + 0 × 3 + 2 × -2 = 8
C(3,1) sums the products of row 3 from A and col 1 from B
= A(3,1) × B(1,1) + A(3,2) × B(2,1) + A(3,3) × B(3,1) + A(3,4) × B(4,1)
= -3 × -1 + 0 × 2 + 2 × -3 + 3 × 0 = -3
C(3,2) sums the products of row 3 from A and col 2 from B
= A(3,1) × B(1,2) + A(3,2) × B(2,2) + A(3,3) × B(3,2) + A(3,4) × B(4,2)
= -3 × 0 + 0 × -4 + 2 × 3 + 3 × -2 = 0

Jason realizes that is a lot of work for even a small example and decides to write a program to help his sister check her calculations.

Input: First line of data file contains the number of data sets which is in range [2,10]. Each data set starts with a line of four whitespace-separated integers: the number of rows r_1 and columns c_1 in the first matrix and the number of rows r_2 and columns c_2 in the second matrix. That line will be followed by r_1 lines, each contains values for c_1 columns, and then r_2 lines, each contains values for c_2 columns. Whitespace separates all values with matrix content values in range [-100,100]. Minimum matrix size is 2 × 2 and maximum matrix size is 10 × 10.

Output: For each data set, display a line that starts with the data set number followed by a colon. If matrix sizes are compatible ($c_1 = r_2$) display, after the colon, the number of rows and columns of the result matrix separated by a comma. If not compatible, display the message "**MATRIX SIZES NOT COMPATIBLE**" after the colon with no extra spacing. When they are compatible, display one line for each row of the result matrix with the column values displayed right aligned in fields exactly 7 characters wide and no additional spacing. Generated values are guaranteed to fit in columns. After each data set, output a line containing exactly 15 carats "**^^^^^^^^^^^^^^^**".

~ ***Input and Output on next page…*** ~
~ ***Jason continued…*** ~

Sample input: (*data aligned for readability*)

```
4
3  4  4  2
  1  7 -3  2
  2 -3  5  2
 -3  0  2  3
 -1  9
  2 -4
 -3  3
  6 -2
3  5  4  3
  7  6 -7  3 -1
 -4  8 -1  8 -3
  2 -1 -6 -6  1
  4  7 -9
  6 -9 -1
 -1  4 -5
  9 -9 -3
5  3  3  4
 -2  6  0
 -1 -1  5
  6 -1 -8
 -3  7  0
  5 -5  0
  1 -3 -6  9
 -9  3 10 -8
  9 -6 -8  5
4  6  5  4
  2 -1 -6 -6  1 -6
  7  6 -7  3 -1  4
  4  5  1  6 -7  4
 -4  8 -1  8 -3 -8
 -1  4 -5  6
  6 -9 -1 -4
  9 -9 -3  0
  8  6 -3  5
  4  7 -9 -3
```

Sample output:

```
1:3x2
     34    -32
    -11     41
     15    -27
^^^^^^^^^^^^^^^
2:MATRIX SIZES NOT COMPATIBLE
^^^^^^^^^^^^^^^
3:5x4
    -56     24     72    -66
     53    -30    -44     24
    -57     27     18     22
    -66     30     88    -83
     50    -30    -80     85
^^^^^^^^^^^^^^^
4:MATRIX SIZES NOT COMPATIBLE
^^^^^^^^^^^^^^^
```

8. Lina

Program Name: Lina.java **Input File: lina.dat**

You and Lina have a Binary Search Tree project due tomorrow, but you haven't started! Quick, write a program to take in a list of strings, make a binary search tree based on that list, and find the diameter, width, height, and number of leaves in the tree. The definitions of these terms are as follows:

- Diameter – The length (number of connections) of the longest path between 2 nodes in the binary search tree.
- Width – The number of nodes in the largest layer of the tree (which level has the most nodes, how many nodes).
- Height – The length (number of connections) of the path from the root of the tree to the lowest node.
- Number of leaves – Leaves are nodes in the tree with no children, find out how many there are.

Duplicate values should be treated as less than and sent to the left subtree.

Input: The input will begin with an integer, `n` (`0 < n <= 1000`), denoting the number of test cases to follow. Each test case will consist of a list of space separated strings of unknown length to be put into the binary search tree. Put the strings into the tree in the order they are given. These strings will not contain any punctuation.

Output: First output the string "`TEST CASE #n:`", where `n` is replaced with the number of the test case, starting at 1. Then, on the following line, output the string "`DIAMETER OF THE TREE: `", followed by the integer diameter of the created binary tree. Then, on the following line, output the string "`WIDTH OF THE TREE: `", followed by the integer width of the created binary tree. Then, on the following line, output the string "`HEIGHT OF THE TREE: `", followed by the integer height of the created binary tree. Then, on the following line, output the string "`NUMBER OF LEAVES IN THE TREE: `", followed by the integer number of leaves in the created binary tree.

Sample input:

```
3
Hello Its Me
I Was Wondering If After All These Years Youd Like To Meet
Somebody Once Told Me The World Was Macaroni
```

Sample output:

```
TEST CASE #1:
DIAMETER OF THE TREE: 2
WIDTH OF THE TREE: 1
HEIGHT OF THE TREE: 2
NUMBER OF LEAVES IN THE TREE: 1
TEST CASE #2:
DIAMETER OF THE TREE: 7
WIDTH OF THE TREE: 3
HEIGHT OF THE TREE: 5
NUMBER OF LEAVES IN THE TREE: 4
TEST CASE #3:
DIAMETER OF THE TREE: 6
WIDTH OF THE TREE: 3
HEIGHT OF THE TREE: 3
NUMBER OF LEAVES IN THE TREE: 3
```

9. Max

Program Name: Max.java **Input File: max.dat**

Max is working on planning a Game Night for him and his friends and has worked out all of the details for the entire night except for one thing: the games! Max, in an attempt to ensure that his friends all have the best time possible during their time together, has already sent out a survey to his friends and received data on which games his friends enjoy the most, and has averaged the score for each game. Max is interested in selecting a set of games which will maximize the enjoyment experienced by his different friends. However, seeing how no one in the groups enjoys only playing a portion of a game, each game selected for the night must be completed in their entirety. However, the group is able to play any given game as many times as they would like, so long as each time they play it, they have enough time to play it to its entirety. Help Max in determining which games he should select for the night given a total amount of time available for the night, as well as a list of games which each have a perceived enjoyment and required playing time.

Input: The first line of input will consist of a single integer n ($1 \leq n \leq 50$) denoting the number of testcases to follow. The next n testcases will consist of three lines. The first line will consist of two space-separated integers. The first of these integers is m ($1 \leq m \leq 100$), denoting the number of games that are to follow, labeled game 1 through game m. The second of these integers is T ($0 \leq T \leq 1440$), denoting the total amount of time that the Game Night will run for. The next line will consist of a string of m space-separated integers denoting the duration ($1 \leq t_i \leq 500$) of the m games. The next line will consist of a single of m space-separated floating-point values (expressed to two decimal places) denoting the perceived enjoyment value ($0 \leq v_i \leq 10$) of the m games. If a duration is the i^{th} element in the space-separated list of integers, then it has a perceived enjoyment value equal to the i^{th} element in the space-separated list of floating-point values.

Output: For each of Max's n requests, on its own line, print the maximum total average perceived happiness obtainable during the Game Night. If this total is equal to 0, then instead print out the string "`Should have picked better games...`". Floating point numbers should be expressed to two decimal places of precision.

Sample input:

```
3
5 300
60 200 15 5 33
6.43 9.31 3.41 2.36 6.54
6 90
200 120 105 115 300 91
7.82 4.50 3.94 5.86 10.00 9.99
2 101
100 1
9.99 0.01
```

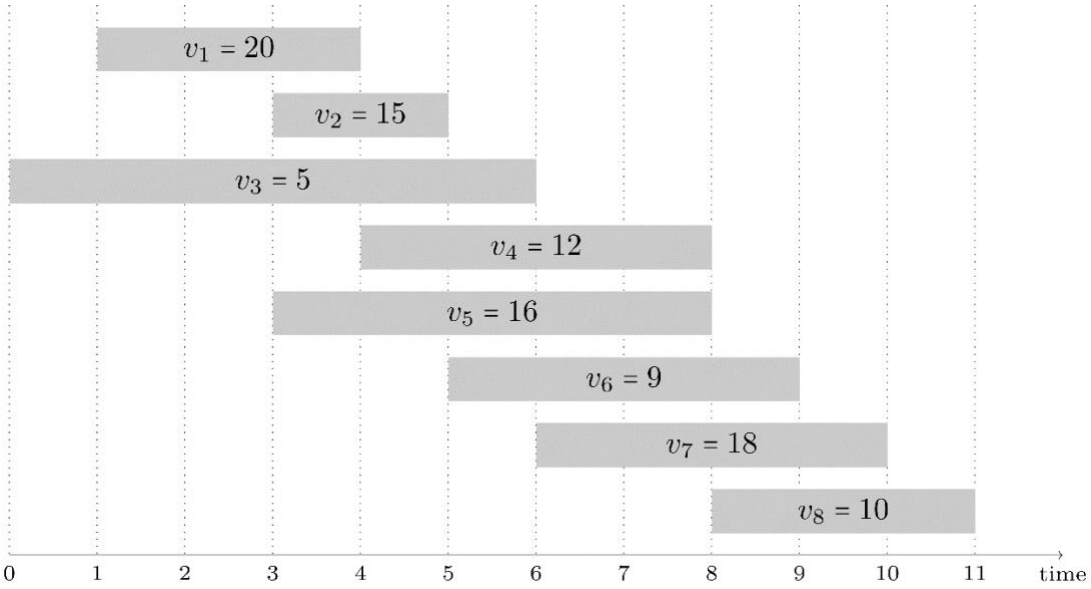
Sample output:

```
141.60
Should have picked better games...
10.00
```

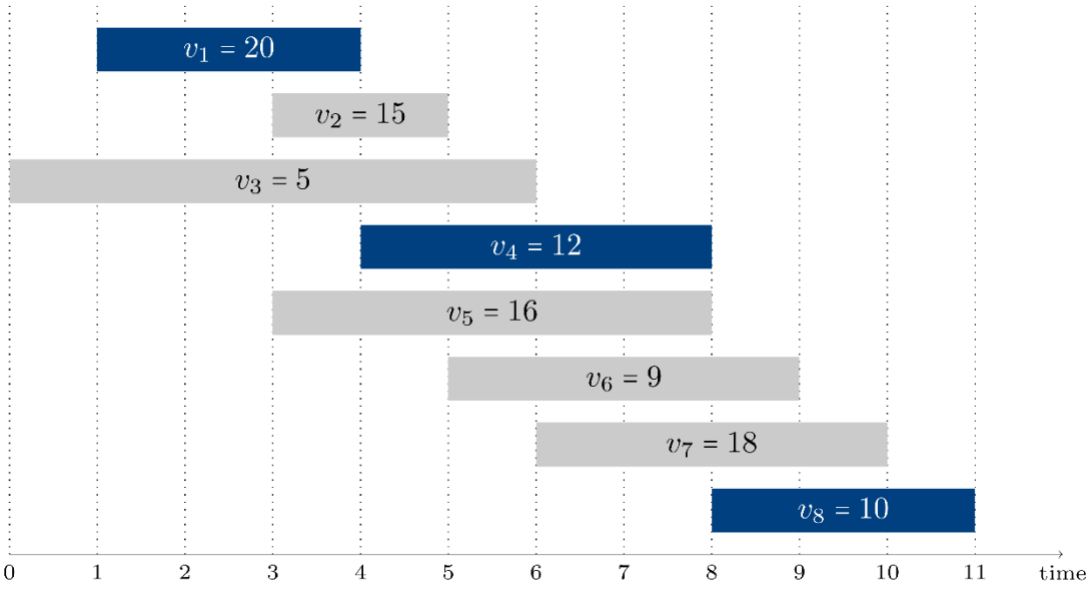
10. Nicholas

Program Name: Nicholas.java **Input File: nicholas.dat**

Nicholas is a highly involved student when it comes to his extracurriculars – so much so that he has accidentally overcommitted himself! Looking to remedy this issue, Nicholas is interested in selecting a subset of the current activities that he is involved in, such that this new subset maximizes his happiness while avoiding the issue of overcommitting himself. However, wanting to be able to devote as much attention as possible to those events, he wishes to those events for their entire duration and is not allowed to leave mid-way through an event.



Because of the sheer quantity of events that he is involved in, as well as the difficult decisions that such an issue might lead to, Nicholas wishes to automate this process of choosing which events to continue participating in. Help Nicholas determine which subset of events that he is currently involved in, he should continue to participate in. The figure below is what such a selection of events would look like, which maximizes the value that Nicholas can obtain from the events described in the figure above, with the selected events being shaded darker than the unselected ones.



Assume that Nicholas has super-human speed and can travel from one event to another in an instant.

~ ***Nicholas continued…*** ~

Input: The first line of input will consist of a single integer n ($1 \leq n \leq 10$) denoting the number of testcases to follow. The next n testcases will consist of two lines. The first line will consist of a single integer, m, denoting the number of events that are to follow. The next line will consist of a single string denoting the list of m start times (s_i), finish times (f_i), and values (v_i) that a particular event (e_i) is composed of. Unique events will be separated by a single comma (the string "`,`"), with the start time, finish time, and value of a particular event each being separated by a single space (the string " "). While not explicitly denoted in the input, events are numbered 1 through m, with 1 being assigned to the first event that appears in the input, 2 being assigned to the second event, and m being assigned to the $last$ event. It is guaranteed that s_i, f_i, and v_i are all integers, as well as $0 \leq s_i < f_i \leq 10^5$, and
$1 \leq v_i \leq 10^4$. Lastly, it should be noted that $1 \leq m \leq 200$.

Output: For each of Nicholas' n requests, on its own line, print the list of events that he should attend which maximize the happiness that attending those events will bring him. This list should be printed as a single line of space-separated integers which represent the event numbers of the events that Nicholas should attend sorted by start time. It is guaranteed that there is a single optimal solution.

Sample input:
```
3
8
1 4 20,3 5 15,0 6 5,4 8 12,3 8 16,5 9 9,6 10 18,8 11 10
9
26 57 29,0 76 31,30 79 39,36 41 9,13 23 34,17 54 71,35 66 5,45 92 46,35 72 8
2
7 56 90,79 100 81
```

Sample output:
```
1 4 8
5 4 8
1 2
```

11. Rufus

Program Name: Rufus.java **Input File: rufus.dat**

You are the new TA for professor Rufus, and she needs you to grade some of her spelling quizzes. For each quiz, you need to find the average score. You will grade by the following criteria:

- Each quiz score will be calculated by finding the least number of "edits" it takes to turn the student's answer into the correct answer. The types of "edits" are as follows:
  - Insert – Insert a new character into the student's answer.
  - Delete – Delete a character from the student's answer.
  - Replace – Replace a character in the student's answer with another character.
- The score for each quiz will be equal to (the length of the correct answer) – (the number of "edits" to turn the student's answer into the correct answer).
- There can be negative scores.
- Ignore case when calculating scores.

Input: The input will begin with two integers, `s`, `q` (`0 < s,q <= 1000`), denoting the number of students who took all of the quizzes, and the number of quizzes to be graded, respectively. There will be `q` cases, and each case will begin with a word, on its own line, denoting the correct answer for that quiz. The following `s` lines will each contain a student's answer to be graded.

Output: Output the average score for each quiz, the average score should then be printed in format: "`[Correct Answer] Quiz Average: [Average Score]`", where `[Correct Answer]` is replaced with the given correct answer, and `[Average Score]` is replaced with the average score on the quiz, rounded to 2 decimal places.

Sample input:

```
3 2
Business
Bizness
Busyness
Bussness
Economics
Ecanomix
Ecenahmecs
Economics
```

Sample output:

```
Business Quiz Average: 6.33
Economics Quiz Average: 6.67
```

12. Victoria

Program Name: Victoria.java **Input File: victoria.dat**

Victoria sometimes gets frustrated while trying to generate good passwords and decided to dig a little deeper into the topic. She discovered that many systems now allow or require passphrases instead of passwords. A passphrase is a collection of "words" separated by spaces and even punctuation marks. The idea is that a phrase of words, even if unrelated, can be remembered more easily than a 10-15 character password of seemingly random characters but no spaces. Longer has always been better when it comes to passwords but computing power has advanced to a point that makes passwords shorter than 10 characters crack-able and passwords longer than 10 characters can be a challenge to remember. Passphrases are typically longer and the addition of spaces and use of all printable characters increases complexity for cracking which makes them stronger.

As with passwords, Victoria suspects that a scoring algorithm might help distinguish between just acceptable passphrases and exceptional passphrases. Here are the rules and scoring algorithm she has proposed and your challenge is to implement and test it.

Rules:

1. A single "word" is a sequence of 1 or more non-blank printable characters and may contain letters, digits, punctuation marks, and all other printable characters. A word could consist of all letters, all digits, all special characters, or any mix of those categories of characters. Uppercase and lowercase letters are different so "UIL" and "Uil" would be distinct words as would be "2024" and "2024!".
2. Passphrases must contain 4 or more distinct words separated by spaces, each word containing at least 4 non-blank characters. "UIL" would not count as a word but "2024" would count.
3. Words with less than 4 non-blank characters, like "UIL", are not included in minimum word count but they do contribute towards overall length and are eligible words for other scoring.
4. Minimum length of an acceptable passphrase is 20 characters including spaces.

Scoring:

1. Acceptable passphrases, those that satisfy ALL of the above rules, receive an **initial** score of 100 points while others receive a **final** score of 0 and are not considered for bonuses and penalties.
2. Each additional word, including ones with length less than 4 characters, beyond the required minimum of 4 words results in a 10-point bonus.
3. Each word that contains a special character (non-letter and non-digit), even if it is a single character or a leading or trailing punctuation mark, results in a 5-point bonus but multiple special characters within the same word do not add more points.
4. Each word that contains a digit results in a 5-point bonus but multiple digits within the same word do not add more points.
5. Each word that contains letters that are only uppercase results in a 10-point bonus but only if less than half of the total words contain letters that are only uppercase. The word may also contain non-letters.
6. A passphrase that contains more than one letter and uses only lowercase letters or only uppercase letters, even with digits and special characters results in a 10-point penalty.
7. Each complete word that matches another complete word results in a 20-point penalty for the matching pair. Test data will not contain a set of 3 or more words that all match but there can be multiple distinct pairs of matching words.
8. Each complete word with a length greater than 2 characters that matches a prefix or suffix or both of another word results in 10-point penalties for each prefix or suffix match. The word itself is not penalized.

Rating Scale:

Score Range	Rating
150 and above	Excellent
125 – 149	Strong
100 – 124	Adequate
75 – 99	Weak
50 – 74	Poor
Below 50	Unacceptable

Examples:

- **"Start with a simple passphrase"** has 4 words of at least 4 characters and 5 total words. It gets a 10-point word bonus for the fifth word with a total score of 110 and would be rated **"Adequate"**
- **"Add some extra digits & special chars!"** has 5 words of at least 4 characters and 7 total words. It gets a 30-point word bonus for the extra 4-char word along with **"Add"** and **"&"**. It also gets a 10-point special-char bonus for **"&"** and "!" for a total score of 140 and would be rated **"Strong"**
- **"Computer Science UIL 2024!!"** has only 3 words of at least 4 characters and is not eligible for any bonuses for a total score of 0 and would be rated **"Unacceptable"**
- **"ing is a prefix of the word ingot and a suffix of confusing and challenging"** has 6 words of at least 4 characters and 15 total words. It gets a 110-point word bonus for extra words. It also gets a 10-point penalty for not having different cases of letters. Words **"a"**, **"of"**, and **"and"** are matching pairs and receive a 60-point penalty. **"ing"** is a prefix of **"ingot"** and gets a 10-point penalty. It is also a suffix of **"confusing"** and **"challenging"** and gets another 20-point penalty. The resulting total score is 110 and would be rated **"Adequate"**

Input: An unknown number of lines with a single passphrase on each line. The length of individual passphrases will not exceed 80 characters. There will be no more than 50 lines in the judge's data file. Lines may contain any of the common ASCII printable characters between and including the space and the tilde. The data will not contain illegal characters.

Output: For each passphrase, output a single line containing its algorithm score followed by a colon (:) and the corresponding passphrase rating.

Sample input:

```
Start with a simple passphrase
Add some extra digits & special chars!
Computer Science UIL 2024!!
ing is a prefix of the word ingot and a suffix of confusing and challenging
AAAAA  12345  *&^%$  zyxwv  q8Z@~  !Mp76
```

Sample output:

```
110:Adequate
140:Strong
0:Unacceptable
110:Adequate
160:Excellent
```

UIL Computer Science Competition

State 2024

JUDGES PACKET - CONFIDENTIAL

I. Instructions

1. The attached printouts of the judge test data are provided for the reference of the contest director and programming judges. Additional copies may be made if needed for this purpose.

2. This packet must remain CONFIDENTIAL. Additional copies may be made and returned to schools when other confidential contest material is returned.

II. Table of Contents

Problem #1
60 Points

1. Ace

Program Name: Ace.java **Input File: None**

Test Input File: None

Test Output To Screen:

```
UIL CompSci State Champion Teams
25 Years Ago - 1999
5A - Houston Langham Creek HS
4A - Fort Worth Dunbar HS
3A - Devine HS
2A - Stockdale HS
1A - Muenster HS
```

Problem #2
60 Points

2. Bart

Program Name: Bart.java **Input File: bart.dat**

Test Input File:

```
13
7 7
$..#.A#
#....##
##....#
@...###
#D....a
###..##
d^...##
4 8
#$##....
J....##B
########
^.j.b..@
3 3
$..
.@.
..^
3 3
$..
...
@.^
8 8
@#A.B.C#
##.D.E.#
#..###.$
#..be#..
#..cd#..
#..#a###
#..#####
#......^
8 8
##A.B.C#
#@.D.E.#
#..###.$
#..be#..
#..cd#..
#..#a###
#..#####
#......^
1 12
$.CBA#@cba.^
1 12
$.CBA@#cba.^
```

~ *Input continues next col* ~

~ *Input continued* ~

```
1 12
$..CBA#@cba^
1 12
$KJI@kji...^
13 10
$..##.#A.B
a.##..C..#
#..D#...b#
E..#..e#@#
##...###.d
F.....G.cH
#####..###
I#J....#K.
f...h#...g
..##...###
#....##j##
i..#.....k
##.^.#####
13 10
$..##.#A.B
a.##..C..#
#..D#...b#
E..#..e###
##...###.d
F.....G.cH
##@##..###
I#J....#K.
f...h#...g
..##...###
#....##j##
i..#.....k
##.^.#####
13 10
$..##.#a.B
A.##..C..#
#..D#...b#
E..#..e###
@#...###.d
F.....G.cH
#####..###
I#J....#K.
f...h#...g
..##...###
#....##j##
i..#.....k
##.^.#####
```

~ *Output on next page* ~

~ ***Bart continued*** ~

Test Output To Screen:

```
Tell Matthew Mcconaughey I said hi.
Exit is short for exciting. 15
Tell Matthew Mcconaughey I said hi.
Exit is short for exciting. 5
Exit is short for exciting. 15
Tell Matthew Mcconaughey I said hi.
Exit is short for exciting. 7
Exit is short for exciting. 8
Exit is short for exciting. 8
Tell Matthew Mcconaughey I said hi.
Exit is short for exciting. 24
Tell Matthew Mcconaughey I said hi.
Exit is short for exciting. 24
```

Problem #3
60 Points

3. Ben

Program Name: Ben.java **Input File: ben.dat**

Test Input File (*indented lines are continuation of previous line*):

```
50
iieegggeeigggppiiiiiinnnnnneeeeeiiiiiiiiiiiiieeepppooooooodddeeedooooddddddnniipppppppp
    ddpppppppggggiiiooooooddddnnnddnggggggggeeeeeeepppnnnnnnpgggggggggddnnnddddddgggiiii
    iiiiiedddddeeeeeeennnnniiiiiggggnnnnnnnndiioiiieennnnnpppppppggggeeeiinnniiiiiiinnnee
    enpppppddddddiiiiiiggggggooooodddddddnnnnnggggggnnnnnggggggnnnnnddddgggggggiiiiiiggggooo
    onnpppppppiiiooooooggggeeggggggggooooooonnnneeeggggggggoopppppppppiiggggggggnniiidgggggggn
    nnnnnnnnnngiinnnnnnddddddddddeeeiiiddpppppppooooooooopppdddooo
ooooooooonniiiiiiippppddddppppppppnnnnnngnnneeeeeeeeddddddooooniiiiiiiiooooiiiiiiiiiinnnnnnnooo
    ooooooooooddddonnnnnppppppppeeeeeeeeoideeenpnnnddddddeeeedgggggggddddiiiiiddeeeeeeeppeeeeee
    nnnnnnnnnnnooooooooeoggggggggddddddddgpppddddddddddddpppppppiiiiipppppppiiiiiieeeiiiiiiiop
    pppppipppiiiiddddddiiiiiiooooogggnnnnpppppppeeeeeeeeggeeeeeigggeeeeeeeeioooooooeeeggggnnnn
    gddddddddeeiiiiiiieeeeeennnnnnnnnnggggggggooooooooiiiiiiigggooooooggggggddeeeeeggggggggdoo
    oonnnnnnnooodddggggggddddddddiiiiiiieeeeggggggeeeeiiiiieeeedgggggnnnggggeeeeeiiiiiiddddddp
    ppppeppppppddddddddeeeeeeeeeennnnnnnooooooopddddddddeddddddooonnnnnooooooeooooeeeeeeggnnn
    nnnneeeeeoooonnnniiieddddddddnneeeeeee
gggpppooooooogggggiiiiiippppppeeeeeeepppiiiieeeoooooiiiiiiiiiiiiiieeeddddddddiiiiiiggppppp
    peeeeeeeeeennnnnneeeeppppppndddddddooooooonnoooooooggggeppppppeeeeooooooeeeeeooooooooooo
    oddggooooooggggggggddddddooooooooiiiiiiinnnniiiiiiiddddddddddddgggggggggiiiiiiieeeedd
    ddddddddeeeeeeeeddddddppppppnnnnnnnppppppiiiiiieeeeppppggggggggggggggooogggggddddnnnnnnnnnn
    nnnneiiiiiiiiiinppp
nnnnggggooooooogiiiiddddgggggnnnnniiiiggggggnneeeoiinooooooooiiiiiiipddddeiiiiipnnnnnnnggg
    ggooooooiddddddggeeeeeeeeppggooooogggggggddddddiiiiiiiddnnnddddeeeeooonnnnnnnggggggggp
    pppppoggddddddiiiiddddddddddiinnddeeeeeoooopppppppggggggeeeeppnnnnnnnoiiiiiiggggggnnnneeee
    gggggggppppppppppppppiiiiiiiooooooooeeeeeeooondddddiiipppppeeddddddddddddooogggggggnnnnnnnonn
    nnnnnooooooeiiiiinniiiiinneeeeennnnnnggggggppppeeeeeeeennnnnnnggggggooooooggggiiiiiooooooooo
    odddddddiiiiggpppooooooopppppppiggggggggddddooooooooiiiiiieeeeeepeeeeeeppppppggggggnnnnpp
    pppppppppppppiggggggggggggdggnnnnnnnnnnnddddddpppppeeooooddddddddddddiiiggggggniiiiiiiiiiiii
    idddddddendddddddeeeeeeeeeeeooooooonppggggggeeeeeeeeeggggggppppppppggggggddddddeeeeeeeooooeep
    pppggggggigddddddgggppppppppddddgggggeeeeeddeeeeeeennnooooooppppiiinnnnnneeggggggooooo
ooooopppppppnnnooooooiiiiiiiiggggggpeeeeddddpppppeeennnnnnnooooooooddddddddiiiiiiiiiiigggggiiiiiiigg
    ggppppppppggggggiiiippppggeeeeeeoooooooeiiiiiooooooooggggggggppppppppdddnnooooooooonnnnnnnnnnnnnn
    ddddddddnppppppppppooooooonnneeeeeeeeppppppppddddiiiiiieeiggggggggddddddddnnnnnppppppppeeeeeniii
    iiippddiiiiiiiippipnnnnnnddddddddddddddddnnnnnnnddgggggggggggggeeeeeeeeeeeeeggggggggggeeeeeeee
    ooooooddggggggnnnnnnnnoonnnnnnnnddddddddiiiddddddppppppiiiiiiieeeppppppoiiippeppppppppiiiiiiieee
    eeoooeeeeeeeeppppppppgggggggggnnnnnnnnppppppggggggggppppipppppppeeeeeeeeeiiooooooooeeeeppppppdddd
    pppooonnnniiiooooooeeeeeeeiiiiippppppddddddnnppppppppnnnggggggggeeeeeennnnnnn
iiiiiiiiieeeeeeooooooonnnnnnnggggggggggggddeeeeeeiiiiieeedddnnnnnnggggoonnnnnnnnnnddddddddooggg
    pnggnnnggggggiiiiiiiiooooiiiiiggggggggooooiiggggpnnneeeennnnnniiiiiiiinnddddiiiiiooiiippppoo
    oooiddddddddddogggggddddddddddnieeeeeeeennnnniiiiiinnnnggggggggddddddppppppppddddddnnnnnnpo
    oeeeeeeeiiiinnnniiiiooooeeepppppddddddeeppppppiiiiiiiggggggnnnnnnddddddnnnniiiiiiieeeeeiigg
    pppppppddddddooooooppppppppddgggggggpnnnnnniieeeeeeeooooooodnnniiiiiiiiggggggeeeennnnggeeeeee
    eddddnnnnnnggggggiiiiiiiiooiiiiiiinnnnnnggggggggeeeeeggggggngggeppppppppeeeeppppppddddggg
    ooogggggeeeeeeeggggggggpiiiiddddddddggggeeeeeeddddddddnnnnnnnnddddooooooiiiiiooppppddggggnnnddd
    deeeeeeeiiiiiidgdooooooeppppppooooooiiiiiiiddddpppppppiiiiieeeeeeeggggggeeennnnnnneeeodii
    iiiiidpddddddnnnnnnnnnddddeeeeeeeiiiiiiiinnennnnnnniiiiiioddddonnnggggggggnnnnnnonppggggdd
    ddennnnnnnnnnnnnneeeeeeeeoonnnnnieeeeeeeeeeeeeeeggggpooooopppppiiinnnniinnnnnnniiiiiiieee
    eeeeeeeeeppnnddddddddiiiiiiiogggggggddddddpppooiiiiiiggggggggeeeeeennnnggggooggggggggnnnn
    nggggggiieeeeepppppgiiiiiggggggggooooooooooooo
```

~ Input file contains 51 lines of data, some lines extremely long – see ben.dat file for complete input ~

~ Output on next page ~

~ Ben continued ~

Test Output To Screen:

```
779320377
565533199
335534016
789931364
207668418
551009422
720607514
850294760
0
0
1630080
0
0
9072
19051200
768623754
810883030
965705837
180430245
400646341
38454136
0
222014579
19179207
910953277
79608138
686774672
310190413
868804412
752722524
368418618
498880570
0
495327522
905195528
296052440
207900
436561971
146428506
369882510
0
4630171
9200736
498715260
465884030
566852933
122285577
102144
29005806
48522726
```

Problem #4
60 Points

4. Bethany

Program Name: Bethany.java **Input File: bethany.dat**

Test Input File:

```
10
10 2
5 6
8 4
2 9
10 8
3 12
5 3
2 20
3 7
2 100
```

Test Output To Screen:

```
2 3 5 7 11 13 17 19 23 29
12 18 20 28 32
6 8 10 14 15 21 22 26
36 100
24 30 40 42 54 56 66 70 78 88
60 72 84
4 9 25 49 121
240 336
64 729 15625
45360 71280
```

Problem #5
60 Points

5. Conrad

Program Name: Conrad.java **Input File: conrad.dat**

Test Input File:

```
10
GAGA
GULP
FOX
ABCDEFGHIJLMNOPQRSTU
DINS
EYE
UT
XXXXX
MUFFIN
NUN
```

Test Output To Screen:

```
GAGA
BAD!!!
FOX
BAD!!!
DINS
EYE
UT
XXXXX
BAD!!!
NUN
```

Problem #6
60 Points

6. Dilbert

Program Name: Dilbert.java **Input File: dilbert.dat**

Test Input File:

```
10 12345 54321 65536 2903185746
17 1 257 2147483647 1073741984 41943185
1717986918 2686456320 2694840320 2863311530 5787213827046133840
1085102592571150095 6510615555426900570 8070450532247928831
4294967296
```

Test Output To Screen: (*indented lines are continuation of previous line*)

```
Test Case: 1: 5 101
Test Case: 2: 9987 10011100000011
Test Case: 3: 35883 1000110000101011
Test Case: 4: 1 1
Test Case: 5: 1251528885 1001010100110001101000010110101
Test Case: 6: 17 10001
Test Case: 7: 1 1
Test Case: 8: 257 100000001
Test Case: 9: 2147483647 11111111111111111111111111111111
Test Case: 10: 41943041 10100000000000000000000001
Test Case: 11: 35913733 10001001000000000000000101
Test Case: 12: 858993459 11001100110011001100110011001100110011
Test Case: 13: 4719621 1001000000010000000101
Test Case: 14: 1285 10100000101
Test Case: 15: 1431655765 10101010101010101010101010101010101
Test Case: 16: 361700864190383365
   1010000010100000101000001010000010100000101000001010000010100000101
Test Case: 17: 1085102592571150095
   1111000011110000111100001111000011110000111100001111000011110000111111
Test Case: 18: 3255307777713450285
   1011010010110100101101001011010010110100101101001011010010110100101101
Test Case: 19: 9223372036854775803
   1111111111111111111111111111111111111111111111111111111111111111111011
Test Case: 20: 1 1
```

Problem #7
60 Points

7. Jason

Program Name: Jason.java **Input File: jason.dat**

Test Input File: ~ *data aligned for readability* ~

```
10
3  4  4  2
  1  7 -3  2
  2 -3  5  2
 -3  0  2  3
 -1  9
  2 -4
 -3  3
  6 -2
3  5  4  3
  7  6 -7  3 -1
 -4  8 -1  8 -3
  2 -1 -6 -6  1
  4  7 -9
  6 -9 -1
 -1  4 -5
  9 -9 -3
5  3  3  4
 -2  6  0
 -1 -1  5
  6 -1 -8
 -3  7  0
  5 -5  0
  1 -3 -6  9
 -9  3 10 -8
  9 -6 -8  5
4  6  5  4
  2 -1 -6 -6  1 -6
  7  6 -7  3 -1  4
  4  5  1  6 -7  4
 -4  8 -1  8 -3 -8
 -1  4 -5  6
  6 -9 -1 -4
  9 -9 -3  0
  8  6 -3  5
  4  7 -9 -3
```

~ *Input continues next column* ~

~ *Input continued* ~

```
10  5  5  7
  2  2  0 -3 -2
  0 -5 -1  6 -2
 -3  0 -7 -6  0
 -6 -5  0 -2 -7
 -1 -3 -2  0  2
  4  0  3 -6 -2
  0 -4  2 -4  2
  8  0 -7 -4 -7
  0  8  3 -8  0
 -6  0 -4 -7 -7
  1  7  3  5 -2  2  0
 -6 -3 -6 -5  0 -2  2
  0 -5  4  6  1 -1  3
  3  0  2 -4  0 -3  0
 -1  3  5  0  5  0 -4
3  2  3  9
 -8 -7
 -7  7
  5  1
  4  3  5  5 -9  6 -5  7  2
  1  0 -5  0 -9 -1 -2 -5 -7
  7 -2  7 -3 -4 -5 -7  7 -8
5  2  2  7
  1 -5
  0 -8
  2  7
 -9  0
 -6 -1
 -9 -2 -8 -7  2  0  0
 -2  1 -1  2 -2 -6  8
```

~ *Input continues next page* ~

~ Jason Input continued ~

```
10 10 10 10
  -61   71    9   54   16  -38 -100  -36  -97  -46
  -63  -35  -55  -33  -71  100  -16  -67   88  -77
    5   50 -100   -6  -75  -85   31    2    2   79
  -95   86   40  -13  -47   88  -72  -23   47  -78
   98   21   -9  -51    1   20  -44   48 -100   25
  -46   56  -95   17   44  -49  -86    3  -19  -15
   13  -37    8  -94   34  -63  -99   -3  -80  -32
   66   70   73   90  -60   42  -84   46  -48   39
   32  -17  -89   35  -51  -35  -19   23  -44  -29
  -31   89  -66   27  -16   16  -73   58  -65  -88
  -50   71   99   90   32   97   25  -41  -80  -97
   84   74   17   70  -69   92   68   76   46   27
  -97  -29   46   89  -76  -79  -78    3  -17  -55
   11  -23   61   51  -45   38   31   67   81   17
   89  -16  -61   96    8   13  -30   82  -60   79
  -17  -70   -6  -40  -58  -40   71   65  -29   57
   28  -76  -29  -29    8  -72  -78  -45   51    5
  -37   73   16  -51  -65  -63  -36  -82  -34    5
   18  -55   74   10  -86   31  -22  -94   19  -17
  -37  -49   -6  -47   65  -24   67   97  -40   95
5 9 9 8
  -15  -51  -20  -86   22   -9    4  -84  -49
  -43   82  -60   47   79  -67   47   18    8
   51   60   28   87  -20   44   94   12  -72
   -7  -82   85   10  -28  -66  -19   -3   62
  -20  -92   46   64  -10    9  -73   40  -13
   86   37   11  -33  -52  -25   59   80
  -35    3   76   71  -38  -35   -7   17
   56   77  -77  -89  -52   74  -17   77
  -96  -52   21  -72  -12   41    2  -79
   75   -2  -97  -62  -91  -86   72   48
   -6  -59   57  -52  -40   56   72    5
   36  -87    8  -62   77  -97  -49   87
   38  -34  -76  -30    5  -81   33  -51
   -9   21  -82   51  -65   75  -67   73
2 2 2 2
   29   87
  -84  -96
  -88   82
   66  -22
```

~ Test Output next page ~

~ Jason continued ~

Test Output To Screen:

```
1:3x2
     34    -32
    -11     41
     15    -27
^^^^^^^^^^^^^^^
2:MATRIX SIZES NOT COMPATIBLE
^^^^^^^^^^^^^^^
3:5x4
    -56     24     72    -66
     53    -30    -44     24
    -57     27     18     22
    -66     30     88    -83
     50    -30    -80     85
^^^^^^^^^^^^^^^
4:MATRIX SIZES NOT COMPATIBLE
^^^^^^^^^^^^^^^
5:10x7
    -17      2    -22     12    -14      9     12
     50     14     28     -5    -11     -7     -5
    -21     14    -49    -33     -1     19    -21
     25    -48    -27      3    -23      4     18
     15     18     17     -2     10      6    -20
    -12      7      2     62    -15     23     17
     10      8     34     48     12     18    -10
      3     70    -47     14    -58     35      7
    -72    -39    -52     10      3      5     25
    -20    -43    -83    -26    -27     13     16
^^^^^^^^^^^^^^^
6:MATRIX SIZES NOT COMPATIBLE
^^^^^^^^^^^^^^^
7:5x7
      1     -7     -3    -17     12     30    -40
     16     -8      8    -16     16     48    -64
    -32      3    -23      0    -10    -42     56
     81     18     72     63    -18      0      0
     56     11     49     40    -10      6     -8
^^^^^^^^^^^^^^^
8:10x10
   9293  14385  -6450  12019   -741  11249   9245  22492   8630   3954
   3627 -15315  -1278 -17134  -8650  -1214   5107 -11302   9266  -1505
   6261   8052    271 -13750  13991  10990  10113  -2866   8470   3129
   4839  -1620   1109   -268 -25102    874   7792   4688  11214   1735
  -8808  19653    847   2837  10267   5868   9968   5385 -18268  -3997
  18849  14503  -8099   1232   4089  17147  11809  11374   3780  10654
  -4388  17254  -9309   4010  14720   2220  -3776  -1809 -15892  -7393
 -15926  14170  19366  12264 -12231   8761  16853  14976  -1949  -5012
    944  13020   2078  -7641   9260   9889   6395  -6691   3654  -6084
  11925  22483  -3302   -588  -8872  12272  12018   5477   6969   1817
^^^^^^^^^^^^^^^
9:5x8
   6728   4190   3480   3723   6221  -2501    891   5253
  -5809  -9147  -1764   4737    356 -20271  -3378  -3167
  -1774 -12955  14984 -17087   4606  -9042   4629   1953
   3008  12527 -18698  -4278  -1380  15193 -11354   5153
  -3863   3405 -10485 -11759  -2650  13978   4506 -14453
^^^^^^^^^^^^^^^
10:2x2
   3190    464
   1056  -4776
^^^^^^^^^^^^^^^
```

Problem #8
60 Points

8. Lina

Program Name: Lina.java **Input File: lina.dat**

Test Input File (*indented lines are continuation of previous line*):

```
15
Hello Its Me
I Was Wondering If After All These Years Youd Like To Meet
Somebody Once Told Me The World Was Macaroni
K A S D E W Q R G H Y U I O P M N B V C X Z L J F T
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
eij weofi wefnn fsverter pwqeork qmler xcvm poiuy lkjh fds rew bvcx gfds mnbv
   lkjh
M N O P Q R S F G H I J K L T U V W X Y Z A B C D E
YUI CVBN RINIM UNVME RIMV UIKMNBG IKMNGTRD ASDFGHJK ZXCVBN QWERTYUI MNBVCXRTYUIK
   OKJGRDCVHJI LKJUIOPOIUH JNBVCFRTREDSXCV
Somebody Once Told Me The World Was Macaroni I Aint The Sharpest Tool In The
   Shed
E F G H I J K L M N O P Q R A B C D S T U V W X Y Z
L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
A B U V W X Y C N O P Q E F G H I R S T Z D J K L M
qwer wert rt rty tyu tyui yui uio iop asd sdf sdf gdfg hfg hghj kjk ljk zxc xcv
   cvb vbn bnm jkl hjk ghj fgh dfg sdf asd qwe wer ert rty tyu yui uio iop
A
B C A
```

~ Test Output on next page ~~

~ Lina continued ~

Test Output To Screen:

```
TEST CASE #1:
DIAMETER OF THE TREE: 2
WIDTH OF THE TREE: 1
HEIGHT OF THE TREE: 2
NUMBER OF LEAVES IN THE TREE: 1
TEST CASE #2:
DIAMETER OF THE TREE: 7
WIDTH OF THE TREE: 3
HEIGHT OF THE TREE: 5
NUMBER OF LEAVES IN THE TREE: 4
TEST CASE #3:
DIAMETER OF THE TREE: 6
WIDTH OF THE TREE: 3
HEIGHT OF THE TREE: 3
NUMBER OF LEAVES IN THE TREE: 3
TEST CASE #4:
DIAMETER OF THE TREE: 12
WIDTH OF THE TREE: 8
HEIGHT OF THE TREE: 7
NUMBER OF LEAVES IN THE TREE: 11
TEST CASE #5:
DIAMETER OF THE TREE: 25
WIDTH OF THE TREE: 1
HEIGHT OF THE TREE: 25
NUMBER OF LEAVES IN THE TREE: 1
TEST CASE #6:
DIAMETER OF THE TREE: 9
WIDTH OF THE TREE: 2
HEIGHT OF THE TREE: 8
NUMBER OF LEAVES IN THE TREE: 6
TEST CASE #7:
DIAMETER OF THE TREE: 20
WIDTH OF THE TREE: 3
HEIGHT OF THE TREE: 13
NUMBER OF LEAVES IN THE TREE: 3
```

~ Output continues next column ~

~ Ouput continued ~

```
TEST CASE #8:
DIAMETER OF THE TREE: 9
WIDTH OF THE TREE: 2
HEIGHT OF THE TREE: 8
NUMBER OF LEAVES IN THE TREE: 5
TEST CASE #9:
DIAMETER OF THE TREE: 9
WIDTH OF THE TREE: 4
HEIGHT OF THE TREE: 5
NUMBER OF LEAVES IN THE TREE: 5
TEST CASE #10:
DIAMETER OF THE TREE: 25
WIDTH OF THE TREE: 2
HEIGHT OF THE TREE: 21
NUMBER OF LEAVES IN THE TREE: 2
TEST CASE #11:
DIAMETER OF THE TREE: 25
WIDTH OF THE TREE: 2
HEIGHT OF THE TREE: 14
NUMBER OF LEAVES IN THE TREE: 2
TEST CASE #12:
DIAMETER OF THE TREE: 13
WIDTH OF THE TREE: 4
HEIGHT OF THE TREE: 13
NUMBER OF LEAVES IN THE TREE: 4
TEST CASE #13:
DIAMETER OF THE TREE: 15
WIDTH OF THE TREE: 7
HEIGHT OF THE TREE: 8
NUMBER OF LEAVES IN THE TREE: 14
TEST CASE #14:
DIAMETER OF THE TREE: 0
WIDTH OF THE TREE: 1
HEIGHT OF THE TREE: 0
NUMBER OF LEAVES IN THE TREE: 1
TEST CASE #15:
DIAMETER OF THE TREE: 2
WIDTH OF THE TREE: 2
HEIGHT OF THE TREE: 1
NUMBER OF LEAVES IN THE TREE: 2
```

Problem #9
60 Points

9. Max

Program Name: Max.java **Input File: max.dat**

Test Input File (*indented lines are continuation of previous line*):

```
50
16 1272
143 361 153 452 465 106 42 381 478 6 43 253 229 474 186 67
2.43 9.06 6.63 3.75 2.32 5.99 5.11 0.76 5.83 5.75 9.13 1.92 8.78 8.83 7.97 5.11
6 477
895 959 525 756 818 655
8.06 7.68 7.31 3.97 9.93 9.17
87 1217
259 312 331 157 451 130 459 44 172 452 78 221 252 443 97 344 100 460 475 158 202
   188 142 365 135 40 150 9 317 109 379 222 391 88 220 314 234 168 320 282 228
   98 56 484 25 175 79 301 215 72 340 266 365 52 180 268 79 203 175 27 322 498
   31 205 388 79 496 124 5 292 326 167 223 149 21 107 315 484 120 198 86 211 14
   494 75 237 329
9.52 8.25 8.46 3.92 2.34 7.09 8.39 1.85 9.90 4.20 1.26 4.15 0.45 2.16 1.01 1.71
   2.14 1.81 5.26 2.88 7.61 6.95 8.69 8.53 3.20 5.53 5.24 0.62 1.67 2.16 9.37
   0.41 0.18 9.10 9.25 4.68 7.63 5.49 7.34 1.02 7.84 7.03 6.00 5.63 7.78 0.91
   8.75 9.58 2.45 3.04 9.79 2.04 0.48 7.50 7.26 1.48 6.97 0.36 8.06 8.26 2.90
   2.53 4.61 4.61 2.41 9.01 6.35 0.49 2.39 4.11 1.35 6.29 0.59 4.97 5.91 5.90
   1.77 8.13 2.49 5.95 3.12 6.22 8.78 0.01 3.31 4.71 3.32
1 1205
84
3.20
8 231
382 251 60 400 51 112 268 281
7.17 6.57 9.02 0.13 2.66 4.63 1.51 0.07
28 1327
750 575 556 597 745 657 906 867 812 622 607 922 701 787 581 600 572 813 636 985
   887 528 855 808 908 986 849 502
3.51 8.20 8.34 7.14 4.25 5.44 8.30 2.34 5.86 6.71 0.94 5.39 2.81 5.38 5.11 2.24
   0.23 4.93 4.86 6.75 9.79 0.71 0.37 7.44 2.47 3.47 2.14 7.61
77 941
188 391 457 60 401 179 107 462 40 225 294 222 431 39 412 71 251 462 192 467 375
   108 167 91 21 378 458 474 14 410 236 131 83 419 385 262 290 35 482 154 247
   356 341 94 462 56 13 171 239 29 228 28 196 154 413 204 97 413 46 357 316 197
   204 273 354 275 324 197 496 27 299 142 280 149 156 460 107
0.78 6.57 2.07 0.02 4.53 2.79 4.21 4.64 6.61 8.33 2.65 7.05 6.71 8.44 6.22 9.62
   1.50 7.37 7.32 5.04 2.38 3.00 1.16 2.89 3.37 6.43 9.72 8.75 9.19 2.40 2.29
   2.97 6.67 8.95 6.53 9.03 4.67 7.42 6.73 3.57 0.38 6.22 8.25 2.80 2.43 4.28
   9.98 2.66 3.58 2.74 8.63 2.35 8.37 5.87 1.67 3.41 3.59 4.60 7.95 9.22 1.93
   8.72 7.57 2.04 1.60 6.20 5.14 7.34 7.39 0.58 3.00 7.66 0.87 9.99 3.58 0.70
   0.21
```

~ Input file contains 151 lines of data, some lines very long – see max.dat file for complete input ~

~ Output on next page ~

~ Max continued ~

Test Output To Screen:

```
1219.00
Should have picked better games...
759.86
44.80
29.72
16.68
718.56
441.45
199.32
1628.90
17.53
4011.72
37.00
9.80
3703.36
8038.66
418.08
1260.00
9.32
399.00
9.91
74.00
566.15
101.73
211.24
476.61
Should have picked better games...
889.53
1133.64
1253.55
850.00
Should have picked better games...
19.18
171.36
136.32
566.74
2787.12
1236.62
108.83
8.22
854.40
1027.20
18.66
345.96
1690.92
40.14
Should have picked better games...
16.52
Should have picked better games...
Should have picked better games...
```

Problem #10
60 Points

10. Nicholas

Program Name: Nicholas.java **Input File: nicholas.dat**

Test Input File: (*indented lines are continuation of previous line*):

```
10
125
54757 86220 4050,60867 68514 8446,18417 52481 1762,37106 97878 1734,1179 41269
   8901,70395 83711 2357,64234 88919 4147,94273 97087 1283,48309 85678
   2935,56430 95762 2268,8255 32150 6082,97591 99809 4421,64778 79358 7899,3846
   93063 4324,44645 93303 2073,27463 49912 1362,82839 92563 6690,91570 97978
   8482,71274 79536 1436,38693 47144 7226,68389 76119 3980,85423 98694
   5351,96279 98968 5380,56618 70340 1690,90954 91735 8034,85353 96633
   1734,85264 91794 4300,39745 72674 8783,71295 93223 9882,27090 96403
   6004,42490 94521 6864,99237 99715 171,68023 86840 5805,52466 65384 5990,20267
   69878 1459,59430 72653 2322,7618 32755 9807,61136 71748 7061,61025 62247
   5507,71796 89391 3413,64751 97156 4395,39376 80659 6576,96183 99124
   9009,57368 68405 9759,61570 89473 561,28898 58107 2703,54909 89009 5497,49593
   97558 3909,75852 90786 1410,50295 71542 969,74267 80091 6332,40957 41077
   7315,28440 39528 8322,31758 40330 654,50001 78125 2916,2434 52358 9746,60370
   78507 3955,14120 84566 5331,24310 39021 6083,44886 48675 5265,9381 74305
   2697,8066 50684 7494,73509 79111 2413,26079 58282 681,54194 97250 6511,68509
   94364 6294,34312 66213 3753,75737 89488 6950,13039 21192 2258,24668 28704
   9900,92763 93641 8665,14365 28687 6975,60308 83394 5801,62285 98343
   9967,54435 95985 2846,79109 99659 4645,62070 68971 8229,96721 96733
   5053,18551 33681 9925,41928 46884 9845,27340 31167 594,2317 85141 2675,92041
   99763 4249,11677 61456 2429,27593 46650 5512,74443 99973 5945,2306 80146
   3466,52453 68257 4896,51931 75932 6869,96072 96450 406,99603 99883 832,25610
   30623 8655,39627 46887 2725,60499 78755 9852,81442 82934 5225,72371 93569
   2404,65497 86988 1273,65991 82616 6475,22292 91059 836,97772 99911 3138,46555
   88608 5433,46670 93235 680,52135 81300 8451,94917 95984 824,90913 93998
   8298,36875 85737 9383,97030 99639 6104,61123 73404 6686,21663 90984
   9255,83493 93590 8461,73860 76568 8665,4483 42335 2371,79672 81589 9109,22444
   59114 5375,19822 89491 3720,20754 99366 3095,39813 84789 2022,21530 94032
   1718,34955 92212 4728,36179 56038 2070,65068 70484 2456,62243 79745
   2672,23142 45692 2092,61955 94286 6200,72270 93773 6671
```

~ Input continues on next page ~

~ Nicholas Input file continued ~ (indented lines are continuation of previous line)

```
176
86979 88213 3354,6465 84421 9054,81672 89202 9195,45884 49069 8508,31836 78840
   3019,64126 82229 8277,8110 73685 7276,73134 97438 8714,35594 61156 5302,82475
   94261 4823,74541 79363 856,83636 92838 10,62215 98677 7778,177 99871
   842,92340 99539 5278,91268 97645 3881,61064 85880 3664,14002 64977 5485,87442
   88258 3113,68782 82098 4185,75424 87144 2473,99525 99937 7006,31036 43600
   6922,41060 53240 9265,70533 78835 9011,86695 92456 7049,45019 66690
   1652,43491 51089 2023,86376 98938 8748,97549 97957 5526,24976 40879
   7239,89933 99061 8282,97865 99446 6687,11258 90187 421,18065 83549 381,32285
   66612 5172,1879 81891 6975,64001 99986 6265,97206 98901 8732,65682 94439
   1114,46833 67563 1515,17798 90202 7823,86334 86537 1004,39688 96652
   7302,75984 90603 7804,43472 44283 1814,14252 99668 2672,76553 92956
   7268,70649 97092 9112,58338 74527 9176,6724 10200 9084,72728 77579 6320,53824
   94087 3979,38980 43903 3511,67978 88785 5820,69279 81183 6226,48363 71517
   6749,93651 98425 7945,53639 86683 155,20720 65947 2553,76974 82445 6601,27356
   46258 2610,97398 99122 7423,69971 97510 2573,43700 78994 9181,65608 87916
   8437,23319 34917 1912,49764 63099 532,48394 51284 2506,20093 53577 9823,54723
   66270 3761,77812 81819 4459,56869 63153 7378,81198 85745 9033,18936 67824
   1064,94351 98422 4472,79990 99227 1955,8327 16668 9003,44960 97194 54,6451
   25886 6755,4892 64650 587,59397 68469 8366,58261 86827 5883,53531 73237
   78,70849 94111 8302,55640 75440 3181,93342 97518 2048,93035 98937 2790,91001
   91812 4655,17054 77981 3916,70688 73289 8018,24039 87366 7801,89191 95735
   3827,27323 61588 3724,19892 79352 2062,19796 22782 1020,48656 95791
   2038,94975 96952 5035,75415 79624 9611,27823 87937 5137,56078 82950
   8408,69817 91401 5841,25428 39309 6827,18111 24851 5501,70720 83255
   1436,50360 95570 4877,57584 59357 4292,16372 36975 8115,53947 74008 223,78953
   83295 6189,18285 65627 6891,49969 80296 1534,59344 72461 7475,66789 67621
   8231,84174 85047 3096,59289 67774 4568,7989 94747 145,20710 52433 811,36070
   99642 9928,86747 88023 351,57110 83850 3365,57903 88830 4755,26437 87691
   8844,49516 71051 4323,13435 53410 4021,93123 94883 4160,41741 63526
   2728,30240 44873 9204,44974 94718 1971,65969 68037 9596,80686 84634
   5073,82639 93646 3907,6798 34888 9515,43048 88838 2436,98296 98561 9186,45846
   59829 5726,17565 91386 3220,81502 96665 5889,4224 71930 8044,92689 98343
   5907,4111 78480 6692,76425 95438 6998,51389 93389 4094,2843 55180 7940,71372
   94001 8278,35096 43895 6132,8209 56359 2888,51873 85805 5588,54204 62462
   4015,25734 52964 6073,74153 91881 5041,41672 66753 3761,66380 93898
   3423,37665 79647 1922,74266 74516 7456,66638 82758 9884,80086 88528
   3953,63299 85097 5520,53534 66675 7218,13150 49740 6390,46700 95649
   9047,70238 94774 4052,86616 88891 9914,27516 63137 6395,6451 30705 4860,74128
   91679 1878,22434 50395 8443,59534 79424 4437,30725 70936 1714,49503 65724
   7451,96552 99166 8644,95139 99917 2489,96384 97533 8321,26947 98636
   3455,66954 91809 4206,32361 66675 8156
13
18388 97622 2026,1745 80115 6104,85145 93327 2989,81558 94721 4529,39279 61319
   711,99703 99932 6305,2876 51755 6699,55766 80806 9201,40265 70096 5092,80828
   81841 2221,5980 62620 9752,42830 46241 510,67509 79180 3
```

~ Input file contains 151 lines of data, some lines very long – see nicholas.dat file for complete input ~

~ Output on next page ~

~ Nicholas continued ~

Test Output To Screen:

```
69 81 54 5 120 108 19 49 17 31 75 26 78 86
78 80 128 4 60 82 109 50 56 17 43 45 48 97 173 30 33 38
7 8 10 1 6
44 34 37 42 17 16 22 41 43 32
60 42 41 69 113 26 107 28 66 64 106 70 72 45 102 5
8 36 34 23 16 3 35 2 25 37
94 37 95 100 59 1 96 24 56 108 21 22 26 33 121 60
23 25 7 26 12 18 8 32 13
62 10 99 116 53 146 128 145 15 97 59 2 52 42 130 102 140 152
25 98 121 16 113 21 24 3 33 40 69 27 123 114 65 67 46
```

Problem #9
60 Points

11. Rufus

Program Name: Rufus.java **Input File: rufus.dat**

Test Input File:

```
7 12
Business
Bizness
Busyness
Bussness
Business
Busstin
Buziness
buzinezz
Economics
Ecanomix
Ecenahmecs
Economics
Economy
Ekinaumiks
Economics
Ekonomiks
ABCDE
ABCD
ABCDF
ABDC
ABCDE
ABCDEF
ABCCE
BCDEF
ABCDE
ABC
AB
ABD
ABCDEFGHIJ
FAC
BCD
ACE
ABCDE
A
B
C
D
E
DF
CW
```

~ *Input continues next column* ~

~ *Input continued from previous column* ~

```
ABCDE
AQ
B
C
DD
EF
WA
CCCCC
ABC
ASDFGHJKL
QWERTYUIO
ZXCVBNMKL
FTYUJNBVF
MNBFTYUIK
QWERTYUIP
LKJHGFDSA
Somebody
Somebodeh
Sumbawdee
somebody
somebodeh
somebodah
Sombodies
Somebodied
Impossible
Iampossible
kimpossible
impossible
impossible
impossible
impossible
impossible
Correct
CORRECT
CoRrEcT
cOrReCt
correct
cORREct
CORRect
correct
```

~ **Input continues on next page** ~

~ Rufus Input continued (previous page contained shorter lines, this one contains long lines) ~
~ (indented lines are continuation of previous line) ~

```
rtyuikjbvcdftyuiknbvcFTYUIKNBVCDRTYIOlkmnbvcdsetyujgtyujvcdftyujBVFTYUJVFDRTYUJN
   BVCDRTYUJHFDRTyytrewsxdewsxdewsxdeSERFCXSERFDXSERFDERFdertyuikllkokjkiokjkJIK
   MJIKMNJIKMJKMJKMNJKIMJ
rtyuikjbvcdftyuiknbvcFTYUIKNBVCDRTYIOlkmnbvcdsetyujgtyujvcdftyujBVFTYUJVFDRTYUJN
   BVCDRTYUJHFDRTyytrewsxdewsxdewsxdeSERFCXSERFDXSERFDERFdertyuikllkokjkiokjkJIK
   MJIKMNJIKMJKMJKMNJKIMJ
rtyuikjbvcdftyuiknbvcyujnbvfyujnbvghjnblkmnbvcdsetyujgtyujvcdftyujBVFTYUJVFDRTYU
   JNBVCDRTYUJHFDRTyytrewsxdewsxdewsxdeSERFCXSERFDXSERFDERFdertyuikllkokjkiokjkJ
   IKMJIKMNJIKMJKMJKMNJKI
rtyuikjbvcdftyuiknbvcFTYUIKNBVCDRTYIOlkmnbvcdsetyujgtyujvcdftyujBVFTYUJVFDRBVCDR
   TYUJHFDRTyytreFCXSERFDXSERFDEwsxdewsxdewsxdeSERRFdertyuikllkokjkiokjkJIKMJIKM
   NJIKMJKMJKMNJKIMJFCXSE
rtyuikjbvcdftyuiknbvcFTYUIKNBVCDRTYIOlkmnbvcdsetyujgtyujvcdftyujBVFTYUJVFDRTYUJN
   BVCDReSERFCXSERFDXSERFDERFdertyuikllkokjkiokjkJIKMJIKMNJIKMJKMJKMNJKIMJTYUJHF
   DRTyytrewsxdewsxdewsxd
rtyuikjbvcdftyuiknbvcFTYUIKNBVCDRTYIOlkmnbvcdsetyujgtyujvcdftyujBVFTYUJVFDRTYURT
   YUJHFDRTyytrewsxdewsxdeSERFCXSFDERFdertyuikllkokjkiokjkJIKMJIKMNJIKMJKMJKMNJK
   IMJkjbvcdftyuiknbvcFTY
rtyuikjbvcdftyuiknbvcFTYUIKNBVCDRTYIOlkmnbvcdsetyujgtyujvcdftyujBVFTYUJVFDRTYUJN
   BVCDRTYUJHFDRTyytrewsxdewsxdewsxdeSERFCXSERFDXSERFDERFdertyuikKIMJ
rtyuikjbvcdftyuiknbvcFTYUIKNBVCDRTYIOlkmnbvcdsetyujgtyujvcdftyujBVFTYUJVFDRTYUJN
   BVCDRTYUJHFDRTyytrewsxdewsxdewsxdeSERFCXSERFDXSERFDERFdertyuikllkokjkiokjkJIK
   MJIKMNJIKMJKMJKMNJKIMJFCXSERFDXSERFDE
qwertyuiopasdfghjklzxcvbnm
qwertyuioplkjhgfdsazxcvbnm
mnbvcxzlkjhgfdsapoiuytrewq
qazxswedcvfrtgbnjhyujmllop
qwertyuioplkjhgfdsazxcvbnm
azxcyuioplkjhvbnmqwertgfds
jhgfdsazxcvbnmqwertyuioplk
qwertyuiopasdfghjklzxcvbnm
```

Test Output To Screen: (*indented lines are continuation of previous line*)

```
Business Quiz Average: 6.00
Economics Quiz Average: 6.57
ABCDE Quiz Average: 3.86
ABCDE Quiz Average: 2.14
ABCDE Quiz Average: 1.00
ABCDE Quiz Average: 0.71
ABC Quiz Average: -5.43
Somebody Quiz Average: 5.29
Impossible Quiz Average: 9.71
Correct Quiz Average: 7.00
rtyuikjbvcdftyuiknbvcFTYUIKNBVCDRTYIOlkmnbvcdsetyujgtyujvcdftyujBVFTYUJVFDRTYUJN
   BVCDRTYUJHFDRTyytrewsxdewsxdewsxdeSERFCXSERFDXSERFDERFdertyuikllkokjkiokjkJIK
   MJIKMNJIKMJKMJKMNJKIMJ Quiz Average: 151.43
qwertyuiopasdfghjklzxcvbnm Quiz Average: 9.71
```

Problem #12
60 Points

12. Victoria

Program Name: Victoria.java **Input File: victoria.dat**

Test Input File:

```
Start with a simple passphrase
Add some extra digits & special chars!
Computer Science UIL 2019!!
ing is a prefix of the word ingot and a suffix of confusing and challenging
AAAAA  12345  *&^%$  zyxwv  q8Z@~  !Mp76
make 25% longer, with special chars
tooo bare forr pass
I really HATE PASS phrases
Needs more words
Let's really SHAKE UP this UIL Contest
Computer Science UIL 2019
how do they come up with this garbage?
A1 && but not OK
bUILding JAVA programs always challenging
longbutnospaces
Java Austin TX May 2019
They drive STUDENTS cra-cra or perhaps CRAY-CRAY (as in computers?), LMAO!!
TxUIL PROBLEM WRITERS are TOTALLY out of their MINDS!
bUILding JAVA programs always challenging
One word is just another word just saying...
One word is just another word, or is it?
IT is what IT is and that isn't what it means!
Pre isn't a prefix or suffix of prewordpre
theprefix comes beforethe the suffix
1234  4123  2134  4123  1
1234  4123  2134  4123  1111
```

Test Output To Screen:

```
110:Adequate
140:Strong
0:Unacceptable
110:Adequate
160:Excellent
125:Strong
0:Unacceptable
110:Adequate
0:Unacceptable
165:Excellent
0:Unacceptable
135:Strong
0:Unacceptable
120:Adequate
0:Unacceptable
0:Unacceptable
225:Excellent
```

~ output continues next column ~

~ output continued from previous column ~

```
195:Excellent
120:Adequate
105:Adequate
130:Strong
140:Strong
145:Strong
80:Weak
0:Unacceptable
115:Adequate
```