Computer Science Competition
District 2023
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	Bogdan
Problem 2	Christine
Problem 3	George
Problem 4	Hisoka
Problem 5	Janice
Problem 6	Krishna
Problem 7	Liza
Problem 8	Miguel
Problem 9	Patrick
Problem 10	Shreya
Problem 11	Sunil
Problem 12	Vanessa

1. Bogdan

Program Name: Bogdan.java **Input File: none**

Bogdan has a big month of January to start the new year in 2023! The month of January starts off with no class on January 2nd as part of the New Year's Celebration. There is also no class on January 16th, in observance of Martin Luther King Day. Last, but certainly not least, Bogdan's 16th birthday is on January 31st.

Bogdan attends the very prestigious Ultimate Intellectual Leadership academy, which utilizes a block scheduling system for their class schedule. Instead of attending every single one of his classes every day for a short period of time, he attends his 1st, 3rd, 5th, and 7th periods on "A" days, and he attends his 2nd, 4th, 6th, and 8th periods on "B" days. Since he only goes to each class every-other-day, each class period is twice as long, as it would have been if all eight periods would be if attended every single day.

With so much going on, Bogdan decides to write a program that will output his calendar in a spreadsheet like format. Can you help him write a program that will output his January calendar so that Bogdan always knows whether he needs to attend his "A" day or "B" day classes?

Input: None

Output: The exact calendar as shown below.

Sample input: None

Sample output:

```
-------------------------------------------------------------
                1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
-------------------------------------------------------------
S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T
U O U E H R A U O U E H R A U O U E H R A U O U E H R A U O U
N N E D U I T N N E D U I T N N E D U I T N N E D U I T N N E
-------------------------------------------------------------
N N A B A B N N A B A B A N N N B A B A N N B A B A B N N A B
o o       o o         o o o       o o         o o   i
    D D D D    D D D D D D     D D D D    D D D D D    D r
C C a a a a C C a a a a a C C C a a a a C C a a a a a C C a t
l l y y y y l l y y y y y l l l y y y y l l y y y y y l l y h
a a       a a         a a a       a a         a a   d
s s       s s         s s s       s s         s s   a
s s       s s         s s s       s s         s s   y
-------------------------------------------------------------
```

2. Christine

Program Name: Christine.java **Input File: christine.dat**

Christine is learning how to use loops in Computer Science class. Her teacher has given her a challenge problem that will demonstrate her skill with using loops. Your job is to create that program so that she may use yours as a guide should she have problems.

Your program should read in exactly six positive integers, each no larger than 32. We will call those integers A, B, C, D, E, and F.

The program will print five lines of output. Each line will consist of from 1 to 32 integers as described below.

Line 1: Print all the integers from A to B.
Line 2: Print all the integers from B to C.
Line 3: Print all the integers from C to D.
Line 4: Print all the integers from D to E.
Line 5: Print all the integers from E to F.

Her teacher gave her three situations to consider:

(1) What if the first number is less than the next.
(2) What if the first number is greater than the next.
(3) What if the two numbers are equal.

Her teacher also told her to look very closely at the sample outputs below.

Input: Input will consist of six integers (A, B, C, D, E, F). Each integer will be in range [1,32].

Output: The five lines of output will consist of all integers with integers separated by one space:
1: from A to B
2: from B to C
3: from C to D
4: from D to E
5: from E to F

Sample input:

```
7
3
3
12
20
5
```

Sample output:

```
7 6 5 4 3
3
3 4 5 6 7 8 9 10 11 12
12 13 14 15 16 17 18 19 20
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5
```

3. George

Program Name: George.java **Input File: george.dat**

Your and your brother George are driving down to see your grandparents for the weekend. Oh No! George forgot to fill up the gas tank! And there are no gas stations between you and your grandparents' house! Quickly find out if you have enough gas to get to their house, or not

Input: The first line will contain a single integer `n (0 < n < 50)` that indicates the number of data sets that follow. Each data set will start with three integers A, B, and C, separated by spaces, denoting the average miles per gallon that your car gets at the current (undisclosed, and unnecessary for the problem) speed, the number of gallons of gas left in the car, and how many miles left between you and your grandparents' house, respectively.

Output: If you have enough gas to get to their house, output the string "`Never tell me the odds.`", otherwise output the string "`I've got a bad feeling about this.`".

Sample input:

```
4
10 15 175
10 15 100
10 10 101
10 10 100
```

Sample output:

```
I've got a bad feeling about this.
Never tell me the odds.
I've got a bad feeling about this.
Never tell me the odds.
```

4. Hisoka

Program Name: Hisoka.java **Input File: hisoka.dat**

Hisoka overheard his parents talking about how they had built a retirement fund by making a regular monthly payment into a retirement annuity. He wants to be able to play with various numbers to see how large the retirement fund can get. Using the formula shown below, it starts with a steady monthly payment into the retirement account. That money earns interest based on a monthly rate that is simply 1/12 of a stated annual percentage rate or APR. A higher APR means more profit from the amount that has been deposited along with monthly interest that has also been added back into the account. The number of periods is just the number of months that payments have been deposited.

Formula:

$$v = p \times \frac{(1+r)^n - 1}{r}$$

Legend:
p = amount of payment
r = monthly interest rate
n = number of months
v = annuity value

Can you help Hisoka create a program to calculate values of savings annuities based on various financial factors?

Input: First line will contain a positive integer **T**, the number of test cases with $1 \leq \mathbf{T} \leq 25$. The following **T** lines will contain three items separated by spaces: amount of a monthly payment, annual percentage rate (APR), and number of years. The monthly payment will be standard dollars and cents in the range \$25.00 … \$1500.00. The APR is a percentage in the range 2.0% … 15.0% and must be converted into decimal form. The number of years will be a whole number in the range 10 … 50.

Output: For each test case, display three items on a single line: sum of all monthly payments into the account, total interest earned, and total value with all items rounded to 2 decimal places and separated by single spaces.

Sample input:

```
3
25.00 12.0 30
50.00 5.75 25
1234.56 7.99 20
```

Sample output:

```
9000.00 78374.10 87374.10
15000.00 18346.73 33346.73
296294.40 429983.64 726278.04
```

5. Janice

Program Name: Janice.java **Input File: janice.dat**

Your best friend Janice and her friend Ariel like to send secret messages to each other, and they've taken to using a Caesar cipher to encode their messages. You need to determine what key they are using, given the encoded and real messages, and use it to decode another message. A Caesar cipher works by shifting the alphabet to the right a given amount (let's call it s), and substituting all the letters in the original message with their corresponding letter in the newly shifted alphabet.

Input: The first line will contain a single integer `n (0 < n < 50)` that indicates the number of data sets that follow. Each data set will consist of three lines, each with their own string of uppercase letters and spaces. The first line will be the actual message that has been sent, the second line will be the encoded version of that message, and the third line is another encoded message to be decoded.

Output: Output the decoded third line of the input, using the first two lines to determine the new alphabet to be used to decode the third line.

Sample input:

```
4
HELLO ITS ME
DAHHK EPO IA
E SWO SKJZANEJC EB WBPAN WHH PDAOA UAWNO UKQZ HEGA PK IAAP
LUKE
SBRL
P HT FVBY MHAOLY
SOMEBODY ONCE TOLD ME
EAYQNAPK AZOQ FAXP YQ
FTQ IADXP IME SAZZM DAXX YQ
ABCDEF
STUVWX
YZABCDEFGHIJKLMNOPQR
```

Sample output:

```
I WAS WONDERING IF AFTER ALL THESE YEARS YOUD LIKE TO MEET
I AM YOUR FATHER
THE WORLD WAS GONNA ROLL ME
GHIJKLMNOPQRSTUVWXYZ
```

6. Krishna

Program Name: Krishna.java **Input File: krishna.dat**

Krishna is intrigued by a song she heard on the radio during December.

Taking the words to the song literally:

On the first day, 1 gift was given. (1)
On the second day, 3 gifts were given (1+2)
On the third day, 6 gifts were given. (1+2+3)
On the fourth day, 10 gifts were given. (1+2+3+4)
On the fifth day, 15 gifts were given. (1+2+3+4+5)

So, if Krishna wanted to know how many gifts were received from Day #3 to Day #5, the answer would be a total of 31 gifts (6+10+15).

Help Krishna write a program to allow her to input two integers A and B representing two different days in the song. The output should be the total number of gifts received from the Day #A to Day #B.

Input: Input will consist of an integer N, the number of test cases. The number of test cases will be in range [1,20]. Each subsequent line will contain two integers A and B representing two different days, both in the range [1,100] where A <= B.

Output: Each line of output will consist of one integer representing the number of gifts received from Day #A to Day #B.

Sample input:

```
5
1 3
1 5
1 12
4 6
10 20
```

Sample output:

```
10
35
364
46
1375
```

7. Liza

Program Name: Liza.java **Input File: liza.dat**

You and Liza have been given detention! It is your job to sort through all recent student assignment grades and print out the top three scores, mean, and median scores for each assignment.

Input: The first line will contain a single integer `n (0 < n < 50)` that indicates the number of data sets that follow. Each data set will begin with a line containing the name of the assignment you are currently working on (this name may contain spaces). The following line will contain an integer `m` (`5 < m < 50`), denoting how many students have grades for this particular assignment. The next `m` lines will contain a students' first and last name, followed by an integer denoting their grade, with no spaces, all separated by commas.

Output: For each data set, output the following: The first line will contain the name of the assignment and nothing more. The next 3 lines will be the names and grades of the top 3 scoring students on the assignment (any ties will be broken alphabetically, so if two students have the same score, the one with the alphabetically first name will be listed first. Sort by last name, then if those are equal, sort by first name. There will never be two students with the same first and last name), in the following format: `<First Name> <Last Name>: <Grade for this assignment>`. After these three lines, you will output the mean score for the assignment, formatted to 2 decimal places, and on its own line, preceded by the string "`MEAN SCORE: `". The last line of output for each case will consist of the string "`MEDIAN SCORE: `" followed by the median score for the assignment, formatted to one decimal place. Each test case should be separated by an empty line, however there should be no extra empty line at the end of the last data set.

Sample input:

```
3
HOMEWORK 1
20
Washington,Bobbi,85
Petrovic,James,84
George,Rahul,64
Gates,Charles,79
Davis,Jack,65
Hatley,Aayush,92
Washington,Monica,89
Targaryen,Aayush,96
Bryant,Angela,70
Chan,Monica,98
James,Benjamin,84
Ketchum,Lebron,86
Jefferson,Angela,68
Jefferson,Cameron,82
Hatley,Samantha,64
Vu,Elsa,96
Parker,Benjamin,93
Petrovic,Phyllis,70
Adams,Henry,73
Lee,Monica,73
```

~ Sample input continues on next page (with no blank line between sections) ~

Liza, continued (with no blank line between sections)

```
EXAM - DIGITAL LOGIC
15
George,Bobbi,83
Anderson,Lebron,94
Franklin,Leia,87
James,Hunter,66
Parker,Tina,98
Adams,David,64
O'neal,Angela,74
Petrovic,Samantha,75
O'neal,David,96
O'neal,Monica,61
Vu,Samantha,99
Davis,Hunter,68
Lee,Samantha,86
Davis,Samantha,77
Gates,Henry,85
MACHINE LEARNING PROJECT PRELIMINARY EVALUATION
16
Galo,Cameron,88
O'reilly,Shaquille,88
Anderson,Benjamin,63
Gates,Elsa,76
Gates,Phyllis,63
Bryant,Hunter,70
Johnson,Tina,70
Vu,Tristan,62
Targaryen,Hunter,89
Vu,Leia,77
Stroud,Phyllis,77
America,Bobbi,100
America,Hunter,67
Hatley,Angela,82
Franklin,Jack,87
Davis,Tina,90
```

Sample output:

```
HOMEWORK 1
Monica Chan: 98
Aayush Targaryen: 96
Elsa Vu: 96
MEAN SCORE: 80.55
MEDIAN SCORE: 83.0

EXAM - DIGITAL LOGIC
Samantha Vu: 99
Tina Parker: 98
David O'neal: 96
MEAN SCORE: 80.87
MEDIAN SCORE: 83.0

MACHINE LEARNING PROJECT PRELIMINARY EVALUATION
Bobbi America: 100
Tina Davis: 90
Hunter Targaryen: 89
MEAN SCORE: 78.06
MEDIAN SCORE: 77.0
```

8. Miguel

Program Name: Miguel.java **Input File: miguel.dat**

Miguel has dreams of creating a very cool and very challenging game involving letters and a puzzle situation. He wants to encrypt a phrase and then have the user try to determine what the original was. Below is an example:

The question might be: What did the Oz girl say to her canine?

The clue would be: `Aaaa, E'ee e eefgiii kl'mn nnn no Ooorrs stttvwy.`

The answer is: `Toto, I've a feeling we're not in Kansas anymore.`

To start this game, Miguel needs a program that will take a quote, or any list of characters, and encrypt it using the following rules:

- If a character is any non-alphabetic character, it will remain in the same position.
- All of the letters will be sorted in alphabetical order not considering the case of the letter.
- Then, the sorted letters will be placed back in the list of characters in positions where letters originally were. If a position originally held an uppercase letter, the new letter in that position will now be uppercase. If a position originally held a lowercase letter, it will again be lowercase.

Look carefully at the example above as you plan your approach.

Miguel is counting on you to write this program so he can start making his millions of dollars as soon as possible.

Input: Input will consist of an integer N, the number of test cases. The number of test cases will be in range [1,20]. Each subsequent line will consist of a list of characters that includes both letters and non-letters. Any printed keyboard character could be used. The length of the string of characters is in the range [1,80]

Output: Each line of output will consist of a list of characters. All non-letters will be in their original positions. Every position that was an uppercase letter will still hold an uppercase letter, and every position that was a lowercase letter will still hold a lowercase letter. Now, however, the original letters have been sorted.

Sample input:

```
5
Dog
University of Texas Interscholastic League
Lmnop1234-5678Qrstabcdefghijk
z-C-N-e=I^q
Rock-Paper-Scissors
```

Sample output:

```
Dgo
Aaacceeeee fg Hiiii Llnnoorrssssttt Tuuvxy
Abcde1234-5678Fghijklmnopqrst
c-E-I-n=Q^z
Acce-Ikoop-Prrrssss
```

9. Patrick

Program Name: Patrick.java **Input File: patrick.dat**

Patrick has been researching mathematical sequences for a special project and stumbled upon one called the "look-and-say" sequence. It is rather unique because it is not generated from an infinite application of a mathematical formula. Instead, the sequence is extended as if a human reader was describing the current number as a count of occurrences of each subsequence containing one specific digit. The starting number, like many numerical sequences is 1. If you were asked to describe the current number it would be one count of the digit one or "one one" which becomes the next number in the sequence 11. The new number contains "two ones" producing 21. Continuing the pattern, "one two, one one" produces 1211 then "one one, one two, two ones" produces 111221. This sequence is shown below along with several more numbers in the sequence.

1, 11, 21, 1211, 111221 as described above and sequence continues as follows:
312211 is "one three, one one, two twos, two ones"
13112221 is "one one, one three, two ones, three twos, one one"
1113213211 is "three ones, one three, one two, one one, one three one two, two ones"
31131211131221 etc.

How about starting with an arbitrary sequence of digits?

7440888 is "one seven, two fours, one zero, three eights"
17241038 is "one one, one seven, one two, one four, one one, one zero, one three, one eight"
1117121411101318 etc.

Can you help Patrick create a program to generated such sequences given an initial starting number?

Input: An unknown number of lines, greater than 1 and no more than 20. Each line contains 2 integers separated by a single space: N, the first number of a sequence with $0 \leq N \leq 999999999999999$ and P, the position of the desired number in the sequence with $1 \leq P \leq 25$. N will not contain a sequence of any single digit that is longer than 9.

Output: For each test case, display one line containing the P^{th} number of the sequence. Length of resulting number will not exceed 2000 digits.

Sample input:
```
1 9
7440888 3
2 13
5 1
```

Sample output:
```
31131211131221
1117121411101318
311311222113111231133211121312211231131112311211232221l2
5
```

10. Shreya

Program Name: Shreya.java **Input File: shreya.dat**

Your friend Shreya left you to pay for lunch, but you only have the coins in your pocket! Quickly, determine whether or not you can pay for the meal given the money you have in your pocket. However, you are a cheapskate, and will only be able to pay for the meal if you can make the target amount **exactly** with the coins in your pocket. For example, if you have 2 coins of value 5, and you need to make 8 to pay for the meal, due to your cheapskate ways, you will not be able to pay for the meal, and you must dine and dash.

Input: The first line will contain a single integer `n (0 < n < 50)` that indicates the number of data sets that follow. Each data set will consist of a line with an unknown number of integers, each denoting the value of one of the coins in your pocket. The following line will contain a single integer denoting the target value, or the value we are trying to make.

Output: If you can make the correct amount, output the string "`Business as usual.`", otherwise, output the string "`Dine and Dash.`"

Sample input:

```
3
2 3 5
6
2 3 3 5
6
4 5 6 7
15
```

Sample output:

```
Dine and Dash.
Business as usual.
Business as usual.
```

11. Sunil

Program Name: Sunil.java **Input File: sunil.dat**

Sunil has been working hard to prepare for this year's UIL programming contests. He feels fairly confident about the written test and most programming techniques but is still unsure of his skills for working with 2-dimension arrays. He wants to build a program that can perform basic processing of arrays of various sizes. Nothing complicated, just basic input and simple output to confirm proper handling of data.

Sunil has requested your assistance on this project, can you help?

Input: First line contains a single integer **T** the number of test cases that follow with $1 \le$ **T** ≤ 10. Each test case starts with a line containing 2 integers separated by whitespace: **R**, the number of rows, and **C**, the number of columns, with both $2 \le$ **R**, **C** ≤ 15. That line will be followed by **R** lines of data with each containing **C** integers separated by whitespace with integers in (-100,100).

Output: For each test case, output four lines. First line contains **R** numbers, one for each row of the array which is the average of data items in that row. Second line contains **C** numbers, one for each column of the array which is the average of data items in that column. Third line contains a single number, the average of all data items. All averages must be rounded to two decimal places and displayed right-aligned in fields that are 8 positions wide with two decimal places. The last line follows previous lines with 25 equal signs "==========" starting in column 1.

Sample input:

```
3
5      2
11     -81
26     86
71     -23
-68    6
-62    48
3      6
40     90     -42    21     97     31
21     -28    -84    67     -85    -67
-30    -55    -36    -99    35     -22
7      7
38     26     -87    76     -29    -34    14
95     35     31     0      -71    -96    -99
-37    -70    -97    -39    56     -36    -27
53     -5     36     4      -36    97     -28
-14    31     17     78     -86    21     -29
16     98     99     93     -19    22     -43
6      -63    -89    99     -81    -41    -41
```

Sample output:

```
  -35.00   56.00   24.00  -31.00   -7.00
   -4.40    7.20
    1.40
==========
   39.50  -29.33  -34.50
   10.33    2.33  -54.00   -3.67   15.67  -19.33
   -8.11
==========
    0.57  -15.00  -35.71   17.29    2.57   38.00  -30.00
   22.43    7.43  -12.86   44.43  -38.00   -9.57  -36.14
   -3.18
==========
```

12. Vanessa

Program Name: Vanessa.java **Input File: vanessa.dat**

Vanessa and her fellow UIL Computer Science teammates, are looking to raise funds to buy new laptops for the upcoming regional and state competitions. Fortunately, a very gracious donor in the community has agreed to make a sizable donation to their cause, but there's a catch… The donor wants Vanessa and her team to earn every dollar received. In order to receive the funding, the donor has devised a game that must be played.

The game the donor has come up with is as follows: The donor provides the team with a set of unique numbers as well as a desired sum to be reached. It is up to the team to determine every possible way to use the unique numbers, that when added up, equal the desired sum. For every valid combination, the donor will donate one dollar to the cause. For example, suppose the donor gave the team the numbers 1, 2, and 3 as well as a desired sum of 4. There are four unique ways this sum can be achieved given the unique number set:

- 1+1+1+1=4
- 1+3=4
- 1+1+2=4
- 2+2=4

In this instance, the team would receive four dollars from the donor and is allowed to play the game again with a different set of numbers and desired sum. Vanessa and her team don't have to list out all the combinations, but instead just have to give the number of possible combinations. If at any point Vanessa and her team are unable to determine the correct number of combinations, the game is over and the donor will cease to distribute any more funds to the team. With so much on the line, Vanessa knows it's in her team's best interest to write a program that will guarantee their submission for the number of combinations to the donor is correct every time, in order to maximize the funds to be received. Can you help Vanessa and her team write such a program?

Input: Input will consist of an integer N, the number of test cases. N will be in range [1,30]. Each test case is made up of three lines. Line one is the unique list of integers in range [1,100]. It will be guaranteed that the list will contain at least 1 integer and no more than 10 integers. The list is separated by commas with no spaces in between. The second line of the test case is the desired sum that is to be reached. The desired sum is guaranteed to be in range [0,2000]. The third line of each test case is 20 dashes meant purely to break up each test.

Output: For each test case, you are to output the exact number of unique combinations that add up to the desired sum. The number of possible combinations will be guaranteed to be in range [0,2147483647].

Sample input:

```
5
1,2,3
4
--------------------
1,2,3,5
5
--------------------
2,5,3,6
10
--------------------
1,2,3,4,5,10
12
--------------------
2,4,6,8
7
--------------------
```

Sample output:

```
4
6
5
49
0
```

UIL Computer Science Competition

District 2023

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. Bogdan

Program Name: Bogdan.java **Input File: none**

Test Input File: None

Test Output To Screen:

```
-------------------------------------------------------------
                  1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
-------------------------------------------------------------
S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T
U O U E H R A U O U E H R A U O U E H R A U O U E H R A U O U
N N E D U I T N N E D U I T N N E D U I T N N E D U I T N N E
-------------------------------------------------------------
N N A B A B N N A B A B A N N N B A B A N N B A B A B N N A B
o o       o o           o o o       o o           o o   i
    D D D D     D D D D D       D D D D     D D D D D     D r
C C a a a a C C a a a a a C C C a a a a C C a a a a a C C a t
l l y y y y l l y y y y y l l l y y y y l l y y y y y l l y h
a a       a a           a a a       a a           a a   d
s s       s s           s s s       s s           s s   a
s s       s s           s s s       s s           s s   y
-------------------------------------------------------------
```

Problem #2
60 Points

2. Christine

Program Name: Christine.java **Input File: christine.dat**

Test Input File:

```
11
21
27
27
19
11
```

Test Output To Screen:

```
11 12 13 14 15 16 17 18 19 20 21
21 22 23 24 25 26 27
27
27 26 25 24 23 22 21 20 19
19 18 17 16 15 14 13 12 11
```

Problem #3
60 Points

3. George

Program Name: George.java **Input File: george.dat**

Test Input File:

```
15
10 15 175
10 15 100
10 10 101
10 10 100
10 10 1000
1 1 1
0 5 5
0 5 0
12 12 145
12 12 144
8 8 64
8 8 65
1 1 10000
1 1 0
4 4 4
```

Test Output To Screen:

```
I've got a bad feeling about this.
Never tell me the odds.
I've got a bad feeling about this.
Never tell me the odds.
I've got a bad feeling about this.
Never tell me the odds.
I've got a bad feeling about this.
Never tell me the odds.
I've got a bad feeling about this.
Never tell me the odds.
Never tell me the odds.
I've got a bad feeling about this.
I've got a bad feeling about this.
Never tell me the odds.
Never tell me the odds.
```

Problem #4
60 Points

4. Hisoka

Program Name: Hisoka.java **Input File: hisoka.dat**

Test Input File:

```
10
25.00 12.0 30
50.00 5.75 25
1234.56 7.99 20
1500.00 15.0 50
1499.99 14.99 10
123.45 2.0 10
100.01 2.09 15
987.65 2.01 29
1111.11 12.11 47
25.01 13.33 33
```

Test Output To Screen:

```
9000.00 78374.10 87374.10
15000.00 18346.73 33346.73
296294.40 429983.64 726278.04
900000.00 206089670.66 206989670.66
179998.80 232573.10 412571.90
14814.00 1570.24 16384.24
18001.80 3119.88 21121.68
343702.20 122322.49 466024.69
626666.04 30979252.10 31605918.14
9903.96 166637.34 176541.30
```

Problem #5
60 Points

5. Janice

Program Name: Janice.java **Input File: janice.dat**

Test Input File:

```
14
HELLO ITS ME
DAHHK EPO IA
E SWO SKJZANEJC EB WBPAN WHH PDAOA UAWNO UKQZ HEGA PK IAAP
LUKE
SBRL
P HT FVBY MHAOLY
SOMEBODY ONCE TOLD ME
EAYQNAPK AZOQ FAXP YQ
FTQ IADXP IME SAZZM DAXX YQ
ABCDEF
STUVWX
YZABCDEFGHIJKLMNOPQR
HOW THE GRINCH STOLE CHRISTMAS
AHP MAX ZKBGVA LMHEX VAKBLMFTL
AX WBWGM
UIL COMPUTER SCIENCE
QEH YKILQPAN OYEAJYA
QEH ZEOPNEYP
UIL REGION
THK QDFHNM
THK RSZSD
UIL INVITATIONAL A
THK HMUHSZSHNMZK Z
THK HMUHSZSHNMZK A
GREEN BEANS
PANNW KNJWB
YXCJCXNB CXVJCXNB
PURPLE ELEPHANTS
GLIGCV VCVGYREKJ
VRK JVVUJ
MACARONI
RFHFWTSN
FSI HMJJXJ
BROCCOLI IS A VICTIM OF A TARGETED MEDIA SMEAR CAMPAIGN
MCZNNZWT TD L GTNETX ZQ L ELCRPEPO XPOTL DXPLC NLXALTRY
DZ LCP MCFDDPWD DACZFED
ABCDEFGHIJKLMNOPQRSTUVWXYZ
HIJKLMNOPQRSTUVWXYZABCDEFG
HIJKLMNOPQRSTUVWXYZABCDEFG
A
A
B
```

Test Output To Screen:

```
I WAS WONDERING IF AFTER ALL THESE YEARS YOUD LIKE TO MEET
I AM YOUR FATHER
THE WORLD WAS GONNA ROLL ME
GHIJKLMNOPQRSTUVWXYZ
HE DIDNT
UIL DISTRICT
UIL STATE
UIL INVITATIONAL B
POTATOES TOMATOES
EAT SEEDS
AND CHEESE
SO ARE BRUSSELS SPROUTS
ABCDEFGHIJKLMNOPQRSTUVWXYZ
B
```

Problem #6
60 Points

6. Krishna

Program Name: Krishna.java **Input File: krishna.dat**

Test Input File:

```
10
1 3
1 5
1 12
4 6
10 20
8 8
1 100
11 13
12 13
5 10
```

Test Output To Screen:

```
10
35
364
46
1375
36
171700
235
169
200
```

Problem #7
60 Points

7. Liza

Program Name: Liza.java **Input File: liza.dat**

Test Input File:

```
15
HOMEWORK 1
20
Washington,Bobbi,85
Petrovic,James,84
George,Rahul,64
Gates,Charles,79
Davis,Jack,65
Hatley,Aayush,92
Washington,Monica,89
Targaryen,Aayush,96
Bryant,Angela,70
Chan,Monica,98
James,Benjamin,84
Ketchum,Lebron,86
Jefferson,Angela,68
Jefferson,Cameron,82
Hatley,Samantha,64
Vu,Elsa,96
Parker,Benjamin,93
Petrovic,Phyllis,70
Adams,Henry,73
Lee,Monica,73
EXAM - DIGITAL LOGIC
15
George,Bobbi,83
Anderson,Lebron,94
Franklin,Leia,87
James,Hunter,66
Parker,Tina,98
Adams,David,64
O'neal,Angela,74
Petrovic,Samantha,75
O'neal,David,96
O'neal,Monica,61
Vu,Samantha,99
Davis,Hunter,68
Lee,Samantha,86
Davis,Samantha,77
Gates,Henry,85
MACHINE LEARNING PROJECT PRELIMINARY EVALUATION
16
Galo,Cameron,88
O'reilly,Shaquille,88
Anderson,Benjamin,63
Gates,Elsa,76
Gates,Phyllis,63
Bryant,Hunter,70
Johnson,Tina,70
Vu,Tristan,62
Targaryen,Hunter,89
Vu,Leia,77
Stroud,Phyllis,77
America,Bobbi,100
America,Hunter,67
Hatley,Angela,82
Franklin,Jack,87
Davis,Tina,90
HW2
9
Ketchum,Lebron,28
Ketchum,Rahul,6
O'neal,Angela,12
Hamilton,Phyllis,61
Chan,Shaquille,70
Targaryen,James,97
Bryant,Grady,12
Jefferson,Samantha,36
Bryant,Samantha,15
AUTOMATA THEORY EXAM 2
46
Franklin,Meredith,70
Jefferson,Phyllis,61
Lee,Meredith,31
Hightower,Samantha,83
Hightower,Angela,4
Parker,David,32
Galo,James,26
Stroud,Henry,60
Gates,Charlie,10
Vu,David,63
Franklin,Monica,57
Petrovic,Darrell,86
Davis,Grady,13
Vu,Hunter,16
George,Angela,18
Hightower,Charles,86
Adams,Darrell,84
Stroud,David,61
O'neal,Tristan,76
Skywalker,Leia,40
Hamilton,Angela,49
Vu,Phyllis,65
America,Elsa,16
Galo,Steven,77
Chan,Tina,94
Targaryen,Lebron,88
```

```
Targaryen,Tina,52
Skywalker,Tina,47
Anderson,Bobbi,3
Stroud,Charlie,57
Bryant,Grady,21
Johnson,Hunter,42
Armstrong,Jack,22
Ketchum,Shaquille,89
Stroud,Rahul,44
Galo,Leia,23
Targaryen,Aayush,62
Bryant,Darrell,49
Davis,Steven,2
Lee,Elsa,6
Gates,Lebron,81
Gates,Bobbi,50
Galo,Aayush,0
Hatley,Phyllis,41
Skywalker,Grady,99
Lee,Samantha,11
PROBABILITY HOMEWORK 9
32
Hightower,Aayush,72
Ketchum,Benjamin,45
O'neal,Phyllis,11
Chan,Elsa,36
Hightower,Charles,38
Chan,Tristan,44
O'neal,Rahul,27
Hightower,James,3
Hatley,Bobbi,13
Skywalker,Monica,27
Targaryen,Lebron,28
Hightower,Henry,42
Parker,Samantha,61
America,Charles,89
Skywalker,Aayush,28
Petrovic,Henry,17
Armstrong,Samantha,10
Bryant,Jack,93
Chan,Jack,86
America,Henry,31
Jefferson,Charlie,59
Hightower,Benjamin,28
Hatley,Lebron,40
Petrovic,Charlie,73
Hamilton,David,33
Bryant,Benjamin,18
James,Elsa,0
Hamilton,Darrell,59
Parker,Darrell,69
Bryant,Hunter,92
George,Janice,31
Bryant,Phyllis,59
MACHINE LEARNING PROJECT REPORT
10
Jefferson,Shaquille,82
Hatley,Darrell,12
Gates,Angela,91
America,Shaquille,96
Chan,Samantha,7
America,Angela,61
Ketchum,Elsa,49
Davis,Steven,85
Washington,Charles,98
Ketchum,Steven,91
MACHINE LEARNING PROJECT FINAL EVALUATION
9
Stroud,Henry,48
Lee,Charlie,51
Chan,David,20
Hightower,Shaquille,46
George,Charlie,99
Lee,Aayush,61
George,Darrell,98
Anderson,James,24
Parker,Benjamin,20
COMPUTER AND NETWORK SECURITY MIDTERM EXAMINATION
14
Jefferson,Jack,57
Targaryen,Hunter,48
America,Leia,2
Hamilton,Rahul,9
George,Samantha,35
Hatley,Hunter,38
Gates,Samantha,45
Hightower,Tina,24
Galo,Aayush,94
Targaryen,Steven,8
Galo,Benjamin,82
Hatley,Benjamin,56
Targaryen,Charlie,64
Vu,Bobbi,40
SOFTWARE ENGINEERING ASSESSMENT
27
George,Phyllis,10
James,Benjamin,1
Parker,Janice,1
Bryant,Samantha,48
Davis,Jack,99
Skywalker,Aayush,33
George,Benjamin,41
Anderson,James,83
Lee,Leia,28
Adams,Benjamin,93
George,Charlie,51
Adams,Samantha,1
Hightower,Elsa,48
Franklin,Tina,55
O'neal,Samantha,78
Hatley,Steven,22
O'neal,Jack,17
O'reilly,Grady,33
Bryant,Henry,23
```

```
Hatley,Leia,58
Hightower,Leia,48
Petrovic,Lebron,4
Lee,Monica,83
O'reilly,Angela,37
O'reilly,Janice,46
Galo,Hunter,38
George,Henry,87
OPERATING SYSTEMS PROJECT CHECKPOINT 2
46
Stroud,Tina,71
Armstrong,Rahul,71
Washington,Tristan,42
Armstrong,Phyllis,72
Petrovic,Phyllis,73
Parker,Rahul,77
Bryant,Samantha,81
Targaryen,Lebron,15
George,Steven,87
O'neal,Lebron,63
Hamilton,Tristan,76
Bryant,Tristan,5
George,Grady,23
O'reilly,Samantha,95
George,Tristan,72
Hatley,Darrell,58
Bryant,Steven,44
Vu,James,25
Skywalker,Grady,33
O'reilly,Cameron,46
O'neal,Charles,66
Bryant,Elsa,60
Franklin,Charles,44
Gates,Lebron,81
Hightower,Aayush,34
Targaryen,Rahul,72
Davis,Elsa,17
Galo,Darrell,5
George,Jack,94
Davis,Jack,25
Vu,Leia,90
Skywalker,Rahul,34
Washington,Meredith,1
Armstrong,Elsa,97
Gates,Cameron,23
America,Steven,4
Adams,Jack,63
Bryant,Charlie,69
Vu,Phyllis,75
Jefferson,Lebron,69
O'reilly,Elsa,96
Vu,Aayush,60
Targaryen,Aayush,20
Armstrong,Benjamin,20
Bryant,James,11
Stroud,Tristan,10
THEATRE PROJECT
39
Lee,Steven,64
Skywalker,Elsa,33
Anderson,Janice,59
Skywalker,Rahul,47
Skywalker,Samantha,7
Hatley,Benjamin,15
Washington,Cameron,62
O'reilly,Tristan,45
Gates,Hunter,79
Washington,Benjamin,54
George,Henry,8
Lee,Bobbi,69
George,Meredith,88
Lee,Hunter,10
Franklin,Charles,83
Galo,Tristan,83
Washington,Charles,91
Hamilton,Charles,53
Targaryen,Darrell,58
Targaryen,Shaquille,0
Adams,Leia,79
Ketchum,David,34
Franklin,Rahul,39
Johnson,Lebron,16
James,Bobbi,88
Chan,Benjamin,26
Stroud,Henry,3
Petrovic,Aayush,56
Washington,Grady,84
Hamilton,Cameron,31
Washington,Samantha,76
Anderson,Tina,95
Gates,Aayush,73
Bryant,Jack,58
Jefferson,Elsa,95
Anderson,Tristan,70
Chan,Steven,59
America,Meredith,59
Bryant,Charlie,29
US GOVERNMENT HOMEWORK 4
29
Armstrong,Charles,64
Lee,Angela,34
O'neal,Cameron,86
Washington,James,0
Ketchum,Janice,29
Skywalker,Rahul,48
Vu,Henry,12
Parker,James,53
Petrovic,Meredith,73
James,Charles,57
Davis,Darrell,71
Petrovic,Bobbi,36
Hamilton,Meredith,27
Vu,Charlie,17
Parker,Elsa,76
```

```
Gates,Charlie,53
Targaryen,David,12
Galo,Elsa,31
Galo,Cameron,87
Parker,Aayush,87
Hatley,Cameron,41
Petrovic,Phyllis,32
O'neal,Jack,65
Jefferson,Hunter,6
Lee,Lebron,46
Chan,Benjamin,18
Lee,Steven,64
Johnson,James,33
Parker,Cameron,67
ELECTRICITY AND MAGNETISM EXAM 3
13
Hightower,Phyllis,17
George,Darrell,73
America,Cameron,98
Bryant,Bobbi,54
Johnson,Tristan,90
Washington,Charlie,45
Gates,Grady,53
Petrovic,Benjamin,72
Bryant,Rahul,41
Galo,James,39
O'neal,Darrell,25
Chan,Darrell,27
Armstrong,Janice,27
MACHINE LEARNING FINAL EXAM
45
Johnson,Elsa,11
O'reilly,Leia,84
Ketchum,Samantha,5
Armstrong,Bobbi,52
Bryant,Shaquille,3
Anderson,Phyllis,85
Targaryen,Janice,22
Gates,Bobbi,57
Franklin,Hunter,22
Hatley,Meredith,15
Davis,Monica,2
Washington,Lebron,67
Bryant,Monica,64
Lee,Darrell,10
Armstrong,Janice,38
Gates,Henry,26
Franklin,Phyllis,72
Johnson,Leia,86
James,Henry,64
Armstrong,Meredith,91
James,Cameron,64
Lee,Shaquille,89
Franklin,Leia,52
Adams,Shaquille,96
Jefferson,Shaquille,48
O'neal,Bobbi,99
James,Bobbi,55
Stroud,Charles,84
Anderson,Hunter,29
Petrovic,Aayush,85
Armstrong,Samantha,18
Ketchum,Charlie,85
Gates,David,26
Armstrong,Jack,97
Hightower,Jack,25
O'reilly,Henry,81
Vu,Grady,30
Stroud,Charlie,74
Targaryen,Samantha,99
Hamilton,Janice,77
Jefferson,Bobbi,30
Hatley,Charles,19
Hatley,Henry,90
Hightower,Bobbi,10
Hamilton,Benjamin,77
```

~ Output continues next page ~

~ Liza, continued ~

Test Output To Screen:

```
HOMEWORK 1
Monica Chan: 98
Aayush Targaryen: 96
Elsa Vu: 96
MEAN SCORE: 80.55
MEDIAN SCORE: 83.0

EXAM - DIGITAL LOGIC
Samantha Vu: 99
Tina Parker: 98
David O'neal: 96
MEAN SCORE: 80.87
MEDIAN SCORE: 83.0

MACHINE LEARNING PROJECT PRELIMINARY EVALUATION
Bobbi America: 100
Tina Davis: 90
Hunter Targaryen: 89
MEAN SCORE: 78.06
MEDIAN SCORE: 77.0

HW2
James Targaryen: 97
Shaquille Chan: 70
Phyllis Hamilton: 61
MEAN SCORE: 37.44
MEDIAN SCORE: 28.0

AUTOMATA THEORY EXAM 2
Grady Skywalker: 99
Tina Chan: 94
Shaquille Ketchum: 89
MEAN SCORE: 47.11
MEDIAN SCORE: 49.0

PROBABILITY HOMEWORK 9
Jack Bryant: 93
Hunter Bryant: 92
Charles America: 89
MEAN SCORE: 42.56
MEDIAN SCORE: 37.0

MACHINE LEARNING PROJECT REPORT
Charles Washington: 98
Shaquille America: 96
Angela Gates: 91
MEAN SCORE: 67.20
MEDIAN SCORE: 83.5

MACHINE LEARNING PROJECT FINAL EVALUATION
Charlie George: 99
Darrell George: 98
Aayush Lee: 61
MEAN SCORE: 51.89
MEDIAN SCORE: 48.0

COMPUTER AND NETWORK SECURITY MIDTERM EXAMINATION
Aayush Galo: 94
Benjamin Galo: 82
Charlie Targaryen: 64
MEAN SCORE: 43.00
MEDIAN SCORE: 42.5

SOFTWARE ENGINEERING ASSESSMENT
Jack Davis: 99
Benjamin Adams: 93
Henry George: 87
MEAN SCORE: 43.19
MEDIAN SCORE: 41.0

OPERATING SYSTEMS PROJECT CHECKPOINT 2
Elsa Armstrong: 97
Elsa O'reilly: 96
Samantha O'reilly: 95
MEAN SCORE: 51.50
MEDIAN SCORE: 60.0

THEATRE PROJECT
Tina Anderson: 95
Elsa Jefferson: 95
Charles Washington: 91
MEAN SCORE: 53.28
MEDIAN SCORE: 58.0

US GOVERNMENT HOMEWORK 4
Cameron Galo: 87
Aayush Parker: 87
Cameron O'neal: 86
MEAN SCORE: 45.69
MEDIAN SCORE: 46.0

ELECTRICITY AND MAGNETISM EXAM 3
Cameron America: 98
Tristan Johnson: 90
Darrell George: 73
MEAN SCORE: 50.85
MEDIAN SCORE: 45.0

MACHINE LEARNING FINAL EXAM
Bobbi O'neal: 99
Samantha Targaryen: 99
Jack Armstrong: 97
MEAN SCORE: 53.67
MEDIAN SCORE: 57.0
```

Problem #8
60 Points

8. Miguel

Program Name: Miguel.java **Input File: miguel.dat**

Test Input File:

```
10
Dog
University of Texas Interscholastic League
Lmnop1234-5678Qrstabcdefghijk
z-C-N-e=I^q
Rock-Paper-Scissors
1!9^2(8)3%7_4=6
Zyxwvutsrqponmlkjihgfedcba
ZAQ!xsw2CDE#vfr4
w
Computer Science Is Awesome
```

Test Output To Screen:

```
Dgo
Aaacceeeee fg Hiiii Llnnoorrssssttt Tuuvxy
Abcde1234-5678Fghijklmnopqrst
c-E-I-n=Q^z
Acce-Ikoop-Prrrssss
1!9^2(8)3%7_4=6
Abcdefghijklmnopqrstuvwxyz
ACD!efq2RSV#wxz4
w
Accceeee Eiimmno Op Rssstuw
```

Problem #9
60 Points

9. Patrick

Program Name: Patrick.java **Input File: patrick.dat**

Test Input File:

```
1 9
7440888 3
2 13
5 1
3 25
4 25
6 25
7 25
8 25
0 25
9 25
13579 11
024680 17
2001 23
1234056789 7
999999999888888 5
333 13
4444 15
010101010101 7
575757575757 10
```

Test Output To Screen: (*indented lines are continuations of long lines*)

```
31131211131221
1117121411101318
3113112221131112311332111213122112311311123112112322211 2
5
3113112221131112311332111213122112311311123112111331121113122112132113121113222112311 3112
   2111213122112311312221121113311211131122211211131221131211132221121321132132212321 12
   1113121112133221123113112221131112212211131221121321131211132221123113112221131112311
   3322112111331121113112221121113122113111231133221121113122113121113222123211211131211
   1213322112132113213221133112132113221321123113213221121113122123211211131221222112112
   3222112311311222113111231133211121321321122111312211312111322211213211321322123211211
   1312111213322112311311222113111231133211121312211231131112311211232221121113311211131
   1222112111312211311123113322112111312211312111322111213122112311311123112112322211211
   1312211312111322212321121113121112131112132112311321322112111312212321121113122122211
   2112322211213211321322113311213212312311211131122211213211321322113221321132132211231
   1311222113311213212322211211131221131211221321123113213221121111312211312113213221132113
   2211221121332211231131122211311123113321112131221123113111231121113311211131221121321
   1331121321132122212211131221131211132222123211211131211112133221121321132132211331121 32
   1132213211231132132211211131221232112111312212221121123222112311311222113111231133211
   1213122112311311123112111331121113122112132113213221132211131221121311121312211213211
   3213221123113112221232112111312211322111312211213211311123113223112111321322123122113
   22212221121123222113
```

~ Output continues on next page ~

~ Patrick, continued ~

```
31131122211311123113321112131221123113111231121113311211131221121321131211132221123113112
    2111213122112311311222112111331121113112221121113122113121113222112132113213221232112
    1113121112133221123113112221131112212211131221121321131211132221123113112221131112311
    3322112111331121113112221121113122113111231133221121113122113121113222123211211131211
    1213322112132113213221133112132113221321123113213221121113122123211211131221222112112
    3222112311311222113111231133211121321321122111312211312111322211213211321322123211211
    1312111213322112311311222113111231133211121312211231131112311211232221121113311211131
    1222112111312211311123113322112111312211312111322111213122112311311123112112322211211
    1312211312111322212321121113121112131112132112311321322112111312212321121113122122211
    2112322211213211321322113311213212312311211131122211213211321322113221321132132211231
    1311222113311213212322211211131221131211221321123113213221121113122113121132211332113
    2211221121332211231131122211311123113321112131221123113111231121113311211131221121321
    1331121321132122212211131221131211132221232112111312111213322112132113213221133112132
    1132213211231132132211211131221232112111312212221121123222112311311222113111231133211
    1213122112311311123112111331121113122112132113213221132211131221121311121312211213211
    3213221123113112221232112111312211322111312211213211311123113223112111321322123122113
    22212221121123222114
31131122211311123113321112131221123113111231121113311211131221121321131211132221123113112
    2111213122112311311222112111331121113112221121113122113121113222112132113213221232112
    1113121112133221123113112221131112212211131221121321131211132221123113112221131112311
    3322112111331121113112221121113122113111231133221121113122113121113222123211211131211
    1213322112132113213221133112132113221321123113213221121113122123211211131221222112112
    3222112311311222113111231133211121321321122111312211312111322211213211321322123211211
    1312111213322112311311222113111231133211121312211231131112311211232221121113311211131
    1222112111312211311123113322112111312211312111322111213122112311311123112112322211211
    1312211312111322212321121113121112131112132112311321322112111312212321121113122122211
    2112322211213211321322113311213212312311211131122211213211321322113221321132132211231
    1311222113311213212322211211131221131211221321123113213221121113122113121132211332113
    2211221121332211231131122211311123113321112131221123113111231121113311211131221121321
    1331121321132122212211131221131211132221232112111312111213322112132113213221133112132
    1132213211231132132211211131221232112111312212221121123222112311311222113111231133211
    1213122112311311123112111331121113122112132113213221132211131221121311121312211213211
    3213221123113112221232112111312211322111312211213211311123113223112111321322123122113
    22212221121123222116
31131122211311123113321112131221123113111231121113311211131221121321131211132221123113112
    2111213122112311311222112111331121113112221121113122113121113222112132113213221232112
    1113121112133221123113112221131112212211131221121321131211132221123113112221131112311
    3322112111331121113112221121113122113111231133221121113122113121113222123211211131211
    1213322112132113213221133112132113221321123113213221121113122123211211131221222112112
    3222112311311222113111231133211121321321122111312211312111322211213211321322123211211
    1312111213322112311311222113111231133211121312211231131112311211232221121113311211131
    1222112111312211311123113322112111312211312111322111213122112311311123112112322211211
    1312211312111322212321121113121112131112132112311321322112111312212321121113122122211
    2112322211213211321322113311213212312311211131122211213211321322113221321132132211231
    1311222113311213212322211211131221131211221321123113213221121113122113121132211332113
    2211221121332211231131122211311123113321112131221123113111231121113311211131221121321
    1331121321132122212211131221131211132221232112111312111213322112132113213221133112132
    1132213211231132132211211131221232112111312212221121123222112311311222113111231133211
    1213122112311311123112111331121113122112132113213221132211131221121311121312211213211
    3213221123113112221232112111312211322111312211213211311123113223112111321322123122113
    22212221121123222117
31131122211311123113321112131221123113111231121113311211131221121321131211132221123113112
    2111213122112311311222112111331121113112221121113122113121113222112132113213221232112
    1113121112133221123113112221131112212211131221121321131211132221123113112221131112311
    3322112111331121113112221121113122113111231133221121113122113121113222123211211131211
    1213322112132113213221133112132113221321123113213221121113122123211211131221222112112
    3222112311311222113111231133211121321321122111312211312111322211213211321322123211211
    1312111213322112311311222113111231133211121312211231131112311211232221121113311211131
    1222112111312211311123113322112111312211312111322111213122112311311123112112322211211
    1312211312111322212321121113121112131112132112311321322112111312212321121113122122211
    2112322211213211321322113311213212312311211131122211213211321322113221321132132211231
```

```
    1311222113311213212322211211131221131211221321123113213221121113122113121132211332113
    2211221121332211231131122211311123113321112131221123113111231121113311211131221121321
    1331121321132122212211131221131211132221232112111312111213322112132113213221133112132
    1132213211231132132211211131221232112111312212221121123222112311311222113111231133211
    1213122112311311123112111331121113122112132113213221132211131221121311121312211213211
    3213221123113112221232112111312211322111312211213211311123113223112111321322123122113
    22212221121123222118
311311222113111231133211121312211231131112311211133112111312211213211312111322211231131122
    2111213122112311311222112111331121113112221121113122113121113222112132113213221232112
    1113121112133221123113112221131112212211131221121321131211132221123113112221131112311
    3322112111331121113112221121113122113111231133221121113122113121113222123211211131211
    1213322112132113213221133112132113221321123113213221121113122123211211131221222112112
    3222112311311222113111231133211121321321122111312211312111322211213211321322123211211
    1312111213322112311311222113111231133211121312211231131112311211232221121113311211131
    1222112111312211311123113322112111312211312111322111213122112311311123112112322211211
    1312211312111322212321121113121112131112132112311321322112111312212321121113122122211
    2112322211213211321322113311213212312311211131122211213211321322113221321132132211231
    1311222113311213212322211211131221131211221321123113213221121113122113121132211332113
    2211221121332211231131122211311123113321112131221123113111231121113311211131221121321
    1331121321132122212211131221131211132221232112111312111213322112132113213221133112132
    1132213211231132132211211131221232112111312212221121123222112311311222113111231133211
    1213122112311311123112111331121113122112132113213221132211131221121311121312211213211
    3213221123113112221232112111312211322111312211213211311123113223112111321322123122113
    222122211211232221110
311311222113111231133211121312211231131112311211133112111312211213211312111322211231131122
    2111213122112311311222112111331121113112221121113122113121113222112132113213221232112
    1113121112133221123113112221131112212211131221121321131211132221123113112221131112311
    3322112111331121113112221121113122113111231133221121113122113121113222123211211131211
    1213322112132113213221133112132113221321123113213221121113122123211211131221222112112
    3222112311311222113111231133211121321321122111312211312111322211213211321322123211211
    1312111213322112311311222113111231133211121312211231131112311211232221121113311211131
    1222112111312211311123113322112111312211312111322111213122112311311123112112322211211
    1312211312111322212321121113121112131112132112311321322112111312212321121113122122211
    2112322211213211321322113311213212312311211131122211213211321322113221321132132211231
    1311222113311213212322211211131221131211221321123113213221121113122113121132211332113
    2211221121332211231131122211311123113321112131221123113111231121113311211131221121321
    1331121321132122212211131221131211132221232112111312111213322112132113213221133112132
    1132213211231132132211211131221232112111312212221121123222112311311222113111231133211
    1213122112311311123112111331121113122112132113213221132211131221121311121312211213211
    3213221123113112221232112111312211322111312211213211311123113223112111321322123122113
    222122211211232221119
111312211312111322212311222122132113213221123113112221133112132123222115132113213221133111
    21321232221171321132132211331121321232221119
132113213221133112132123123112111311222112132113311213211231232112311311222112111312211312
    1123113322112132113212231121113112221121321132122211322212221121123222110132113213221
    1331121321231231121113112221121321133112132112312321123113112221121113122113111231133
    2211213211321223112111311222112132113212221132221222112112322211213211321322113311213
    2123123112111311222112132113311213211231232112311311222112111312211311123113322112132
    1132122311211131122211213211321222113222122211211232221141321132132211331121321231231
    1211131122211213211331121321123123211231131122211211131221131112311332211213211321223
    1121113112221121321132122211322212221121123222116132113213221133112132123123112111311
    2221121321133112132112312321123113112221121113122113111231133221121321132122311211131
    1222112132113212221132221222112112322211813211321322113311213212312311211131122211213
    2113311213211231232112311311222112111312211311123113322112132113212231121113112221121
    321132122211322212221121123222110
```

~ Output continues on next page ~

~ Patrick, continued ~

```
13211321322113311213212312311211131122211213211331121321123123211231131122211211131221131
    1123113322112132113213221133122112231131122211211131221131112311332211211131221131211
    1322212321121113121112133221121321132132211331121321132213211231132132211211131221232
    1121113122122211211232221123113112221131112311332111213122112311311123112111331121113
    1221121321131211132221123113112211121312211231131122211211133112111311222112111312211
    3121113222112132113213221133112132113311211131221222112111322132112311311222123222113
    3122211311221122311311222113111231133211121312211231131112311211133112111312211213211
    3121113222112311311221112131221123113112221121113311211131122211211131221131211132221
    1213211321322123211211131211121332211231131122211311122122111312211213211312111322211
    2311311222113111231133221121113311211131122211211131221131112311332211211131221131211
    1322212321121113121112133221121321132132211331121321132213211231132132211211131221232
    1121113122122211211232221123113112221131112311332111213213211221113122113121113222112
    1321132132212321121113121112133221123113112221131112311332111213211322111213111213211
    2311312111322111213112221133211322112211213322110111312211312111322212321121113121112
    1311121321123113213221121113122123211211131221121311121312211213211321322112311311222
    1133112132123222112111312211312111322212311222122132113213221123113112221133112132123
    2221123113112221131112311332111213122112311311123112112322211211131221131211132221232
    1121113122113221113122112132113121113222112311311221112131221123113112211322112211213
    3221121321132132211331121321231231121113112221121321133112132112312321123113112221121
    1131221131112311332211213211321223112111311222112132113213221123123211231132132211231
    1311222113111231133221121113122113121113222123211211131221232112311311221132211231132
    21113122112132113213211121332212311322113212221
13211321322112311311123112111311222114311311222110311311222115311311222116311311222117311
    311222118311311222119
1112111931163118
132113213221133112132123222112132113213221133112132123222113
31131122211311123113321112131221123113111231121123222112311311222113111231133211121312211
    23113111231121123222114
31131122211013211321322110132113213221101321132132211013211321322110132113213221101311222
    1
31131122211311123113322115311311222113111231133221173113112221131112311332211531131122211
    3111231133221173113112221131112311332211531131122211311123113322117311311222113111231
    1332211531131122211311123113322117311311222113111231133221153113112221131112311332211
    73113112221131112311332211531131122211311123113322117
```

Problem #10
60 Points

10. Shreya

Program Name: Shreya.java **Input File: shreya.dat**

Test Input File:

```
12
2 3 5
6
2 3 3 5
6
4 5 6 7
15
1 2 3 4 5
15
1 2 3 4 5
16
1 2 3 4 5 6
16
1 1 1 2 2 2
8
3 2 3 2 3 2
14
3 2 3 2 3 2
12
1 1 1 1 1 1 1 1 1 1
10
1 1 1 1 1 1 1 1 1 1
11
1 1 1 1 1 1 1 1 1 1
12
```

Test Output To Screen:

```
Dine and Dash.
Business as usual.
Business as usual.
Business as usual.
Dine and Dash.
Business as usual.
Business as usual.
Dine and Dash.
Business as usual.
Business as usual.
Dine and Dash.
Dine and Dash.
```

Problem #11
60 Points

11. Sunil

Program Name: Sunil.java **Input File: sunil.dat**

Test Input File: (*rows of data indented and right-aligned here for readability, actual data single tab delimited*)

```
7
5   2
   11  -81
   26   86
   71  -23
  -68    6
  -62   48
3   6
   40   90  -42   21   97   31
   21  -28  -84   67  -85  -67
  -30  -55  -36  -99   35  -22
7   7
   38   26  -87   76  -29  -34   14
   95   35   31    0  -71  -96  -99
  -37  -70  -97  -39   56  -36  -27
   53   -5   36    4  -36   97  -28
  -14   31   17   78  -86   21  -29
   16   98   99   93  -19   22  -43
    6  -63  -89   99  -81  -41  -41
2   2
  -29   10
   84   61
10  10
  -73  -75  -73   48  -50   23   39  -46  -55  -50
  -90   -7  -15  -90   68   91   50   19   42   -6
   -3  -58   38   -2   78   20   20  -48    3   97
   65   99   49  -79   49  -49   -1  -54   50   71
   18   19  -34    0  -54  -50  -51   77  -94  -93
   41  -93  -20  -39   93  -92   86  -62   62   70
  -13   39   -5   44   88    4   13   -8   71   27
  -21  -47   97   21  -87   77   16  -66  -84  -65
  -23  -13    3   25    6  -70   73   67   31   35
  -86  -99  -56  -37   -9  -40  -90   15   79  -53
2   7
    5   -2  -51  -11    4  -65  -88
  -53   81   86   90   46    1  -87
3   3
  -25   26   13
   81   91  -53
  -48  -95  -26
```

~ Output continues on next page ~

~ Sunil, continued ~

Test Output To Screen:

```
  -35.00   56.00   24.00  -31.00   -7.00
   -4.40    7.20
    1.40
=========================
   39.50  -29.33  -34.50
   10.33    2.33  -54.00   -3.67   15.67  -19.33
   -8.11
=========================
    0.57  -15.00  -35.71   17.29    2.57   38.00  -30.00
   22.43    7.43  -12.86   44.43  -38.00   -9.57  -36.14
   -3.18
=========================
   -9.50   72.50
   27.50   35.50
   31.50
=========================
  -31.20    6.20   14.50   20.00  -26.20    4.60   26.00  -15.90   13.40  -37.60
  -18.50  -23.50   -1.60  -10.90   18.20   -8.60   15.50  -10.60   10.50    3.30
   -2.62
=========================
  -29.71   23.43
  -24.00   39.50   17.50   39.50   25.00  -32.00  -87.50
   -3.14
=========================
    4.67   39.67  -56.33
    2.67    7.33  -22.00
   -4.00
```

Problem #12
60 Points

12. Vanessa

Program Name: Vanessa.java **Input File: vanessa.dat**

Test Input File:

```
6
1,2,3,4,5,6,7,8,9,10
45
--------------------
5,10,15,20
200
--------------------
2,4,6,8,20,25,35
150
--------------------
10,20,30,40,50,60,70,80,90,100
1250
--------------------
1,3,5,9
7
--------------------
1,3,5,9
1
--------------------
```

Test Output To Screen:

```
33401
632
14264
32220069
4
1
```