IEEEXtreme 5.0
Problem Set

Gregory A. Lussier
IEEE Student Branch
École de technologie supérieure
Montréal, Canada

November 1, 2011

Abstract

This document contains the problem set of the 2011 edition of the IEEEXtreme coding competition.

Contents

1 Question A
Computing Pearson's correlation coefficient **5**
1.1 Description . . . 5
1.2 Task . . . 5
1.3 Input . . . 5
1.4 Output . . . 5
1.5 Sample Input . . . 5
1.6 Sample Output . . . 5

2 Question B
Implementing a letter-specific Bloom filter **6**
2.1 Description . . . 6
2.2 Task . . . 6
2.3 Input . . . 6
2.4 Output . . . 6
2.5 Sample Input . . . 6
2.6 Sample Output . . . 6

3 Question C:
Longest Cable Way **7**
3.1 Description . . . 7
3.2 Task . . . 7
3.3 Time Limit . . . 7

3.4 Input . . . 7
3.5 Output . . . 8
3.6 Note . . . 8
3.7 Sample Input - Scenario 1 . . . 8
3.8 Sample Output - Scenario 1 . . . 8
3.9 Sample Input - Scenario 2 . . . 8
3.10 Sample Output - Scenario 2 . . . 8

4 Question D: Passing the Number Ball **9**
4.1 Description . . . 9
4.2 Task . . . 10
4.3 Time Limit . . . 10
4.4 Input . . . 10
4.5 Output . . . 11
4.6 Sample Input . . . 11
4.7 Sample Output . . . 11

5 Question E: Kakuro Validation **12**
5.1 Description . . . 12
5.2 Task . . . 13
5.3 Time Limit . . . 13
5.4 Input . . . 13
5.5 Output . . . 14
5.6 Sample Input - Scenario 1 . . . 14
5.7 Sample Output - Scenario 1 . . . 15
5.8 Explanation - Scenario 1 . . . 15
5.9 Sample Input - Scenario 2 . . . 16
5.10 Sample Output - Scenario 2 . . . 16
5.11 Explanation - Scenario 2 . . . 17
5.12 Sample Input - Scenario 3 . . . 17
5.13 Sample Output - Scenario 3 . . . 17
5.14 Explanation - Scenario 3 . . . 17
5.15 Sample Input - Scenario 4 . . . 17
5.16 Sample Output - Scenario 4 . . . 18
5.17 Explanation - Scenario 4 . . . 18
5.18 Sample Input - Scenario 5 . . . 18
5.19 Sample Output - Scenario 5 . . . 19
5.20 Explanation - Scenario 5 . . . 19

6 Question G: Which integer is closer to “hello”? **20**
6.1 Description . . . 20
6.2 Task . . . 21
6.3 Input . . . 21
6.4 Output . . . 22

6.5 Sample Input 1 . . . 22
6.6 Sample Output 1 . . . 22

7 Question H:
Tracking billiard balls **23**
7.1 Description . . . 23
7.2 Task . . . 23
7.3 Input . . . 23
7.4 Output . . . 23
7.5 Sample Input . . . 23
7.6 Sample Output . . . 24

8 Question I:
Help Bugs Bunny escape Elmer **25**
8.1 Description . . . 25
8.2 Task . . . 25
8.3 Input . . . 25
8.4 Output . . . 25
8.5 Sample Input . . . 26
8.6 Sample Output . . . 26

9 Question J:
Resistors Network **27**
9.1 Task . . . 27
9.2 Input . . . 27
9.3 Output . . . 27
9.4 Sample Input . . . 27
9.5 Sample Output . . . 27

10 Question N:
Islands **28**
10.1 Description . . . 28
10.2 Task . . . 28
10.3 Input . . . 28
10.4 Output . . . 28
10.5 Input 1 . . . 28
10.6 Output 1 . . . 29
10.7 Input 2 . . . 29
10.8 Output 2 . . . 29
10.9 Input 3 . . . 29
10.10 Output 3 . . . 29

11 Question O:
Patron **30**
11.1 Description . . . 30
11.2 Task . . . 30
11.3 Input . . . 30
11.4 Output . . . 30

11.5 Input 30
11.6 Output 31
11.7 Explanation 31

12 Question S:
Simple polygons **32**
12.1 Description 32
12.2 Task 32
12.3 Input/Output format 32
12.4 (Isosceles triangle) Input: 32
12.5 (Isosceles triangle) Expected output 32

13 Question V:
Hogwart's Cup **33**
13.1 Description 33
13.2 Input Format 33
13.3 Output Format 33
13.4 Test Input 33
13.5 Test Output 33

14 Question W:
Maya Diet **34**
14.1 Description 34
14.2 Inputs 34
14.3 Output 34
14.4 Example Input 1 34
14.5 Example Output 1 35
14.6 Example Input 2 35
14.7 Example Output 2 35

15 Question X:
Hacker's Challenge **36**
15.1 Description 36
15.2 Input 36
15.3 Output 36
15.4 Example Input 36
15.5 Example Output 36

1 Question A
Computing Pearson's correlation coefficient

1.1 Description

Pearson's correlation coefficient is a statistical measure of correlation between two data sets or series. The value of the coefficient ranges from +1 (perfect positive correlation) to -1 (perfect negative correlation). Correlations can indicate a predictive relationship that can be useful in practical applications. The formula for Pearson's correlation coefficient for two series X and Y is:

$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$

where μ_x is the population mean of series X (and μ_x is the same for series Y), μ_x is the population standard deviation of series X (and μ_x the same for series Y), and $E[]$ is expected value.

1.2 Task

Write a program that accepts as input two series of values, X and Y, and computes the Pearson correlation coefficient (). You may assume that each series has the same number of values.

1.3 Input

Series of values X and Y seperated by space.

1.4 Output

The program should output the value of $\rho_{x,y}$ to four decimal places. If $\rho_{x,y}$ cannot be computed, then the output should be the string 'invalid input'. Note: The input ends with a delimiter. There is no new line character at the end of the result.

1.5 Sample Input

```
14 2
16 5
27 7
42 9
39 10
50 13
83 20
-
```

1.6 Sample Output

The following is the output for the above sample input.

```
0.9856
```


2 Question B
Implementing a letter-specific Bloom filter

2.1 Description

A Bloom filter is a space-efficient data structure used for probabilistic set membership testing. When testing an object for set membership, a Bloom filter may give a false positive (that is, return a "true" for the tested object when the object is not mapped into the filter), but never a false negative (that is, return a "false" when the tested object is mapped into the filter). For this problem you will implement a Bloom filter specifically intended for mapping and testing of English words.

2.2 Task

Write a program accepts as input a series of mapping words (say, "cat", "dog", and "bird") to map or load the Bloom filter as described below. Once the Bloom filter is mapped the program should accept words to test against the mapped filter. The program should return TRUE or FALSE for each entered test word as described below. All inputs are case insensitive (that is, "Cat" and "cat" should be considered to be the same). The Bloom filter is a boolean array indexed from 1 to 26. The filter array is initialized to all array values, or positions, set to FALSE. Each letter of a mapping word sets the corresponding alphabetical position value in the array to TRUE (if it is already TRUE from a previous mapping, then it remains TRUE). For example, "Cat" would set array positions 3, 1, and 20 to TRUE corresponding to position 3 for "C", 1 for "a", and 20 for "t". When testing for set membership, each position in the filter array must be TRUE for the positional value of each letter in the test word for the program to return TRUE, else FALSE is returned.

2.3 Input

The first line contains a series of English words delimited by spaces, line breaks, and/or punctuation ".", ",", "?", and "!". The second line contains the testing words.

2.4 Output

The program should output the number of test words (from the testing file) that tested as TRUE in the Bloom filter.

2.5 Sample Input

cat dog bird turtle
Bottle MOUSE cat doggy Dog

2.6 Sample Output

For the above sample input the output should be:
3

3 Question C: Longest Cable Way

3.1 Description

Historically, all cableways have been less than 100 kilometers in length and cable car speeds have not crossed 50 kmph. However, the discovery a new ore material, available directly in the form of long rods, in the Grand Canyon, led to the ability to create very high speed cable ways that can extend to thousands of kilometers. The US army decided to connect most of their air bases and navy bases using long-distance cable ways. However, there were a few constraints.

1. The cables were in the form of long and flexible rods with smooth edges that can be joined together using a very expensive but specialized grafting process. However, cutting the cables was not possible as it was not possible to retain the smoothness of the edges that is required to graft them.

2. The join process was not only very expensive but also inefficient as the speeds of the cable cars were affected by the joints and reduced by nearly 5% for every joint. Hence, there was a need to minimize the number of joints.

3. The cables, being naturally available rods of varying lengths, were available in specific sizes and there was limited inventory that was available.

4. The cable way should be connected using cable that is the exact length of the distance between the two locations (no higher and no lesser).

3.2 Task

Now your task is to help the engineer select the optimal combination of cables from the inventory to build the exact requested length of cable way such that the number of joints is minimized. Write a program to achieve the same. Your program must output only the minimum number of joints possible.

3.3 Time Limit

Maximum five seconds for any combination of inventory and up to 5000 kilometers of cable way.

3.4 Input

The first line of the input D is the distance between the two air bases to be connected in kilometers. (0 < D < 5000)

Each subsequent line contains a pair of numbers Li Ni, indicating the length of the cable in kilometers and the quantity available in the inventory. ($0 < i <= 20, 1 <= Li <= 200, 1 <= Ni <= 100$)

The input is terminated by 0 0.

3.5 Output

1. The output should contain a single integer J representing the minimum number of joints possible to build the requested length of the cable way

2. If no solution is possible the program should print “No solution possible” in the output.

3.6 Note

There is no new line character at the end of the result. There may be multiple solutions for the same number of minimum joints.

It is sufficient to identify the number of joints and not the actual list of rods to be used.

Example: (see below for input formats)

Solutions for making 444 with 10 joints are

$2 * 2 + 50 * 7 + 45 * 2 = 11$ rods of cables i.e., 10 Joints

$16 * 1 + 3 * 1 + 30 * 1 + 50 * 7 + 45 * 1 = 11$ rods of cables i.e., 10 Joints

However the following has 11 joints and hence not a solution

$3 * 2 + 30 * 1 + 50 * 8 + 8 * 1 = 12$ rods of cables i.e., 11 Joints

3.7 Sample Input - Scenario 1

```
444
16 2
3 2
2 2
30 3
50 10
45 12
8 12
0 0
```

3.8 Sample Output - Scenario 1

```
10
```

3.9 Sample Input - Scenario 2

```
44
30 31
50 10
45 12
90 21
43 1
0 0
```

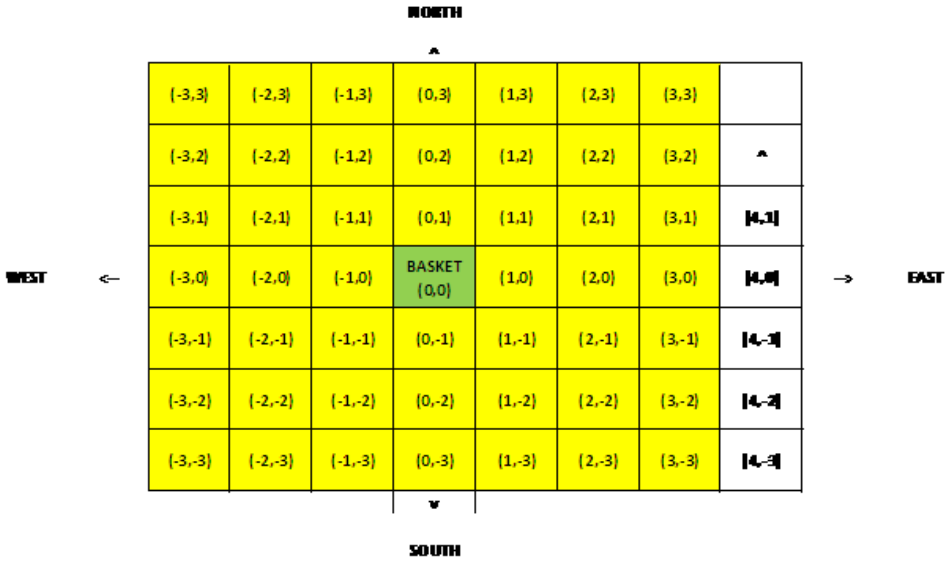
3.10 Sample Output - Scenario 2

```
No solution possible
```


4 Question D: Passing the Number Ball

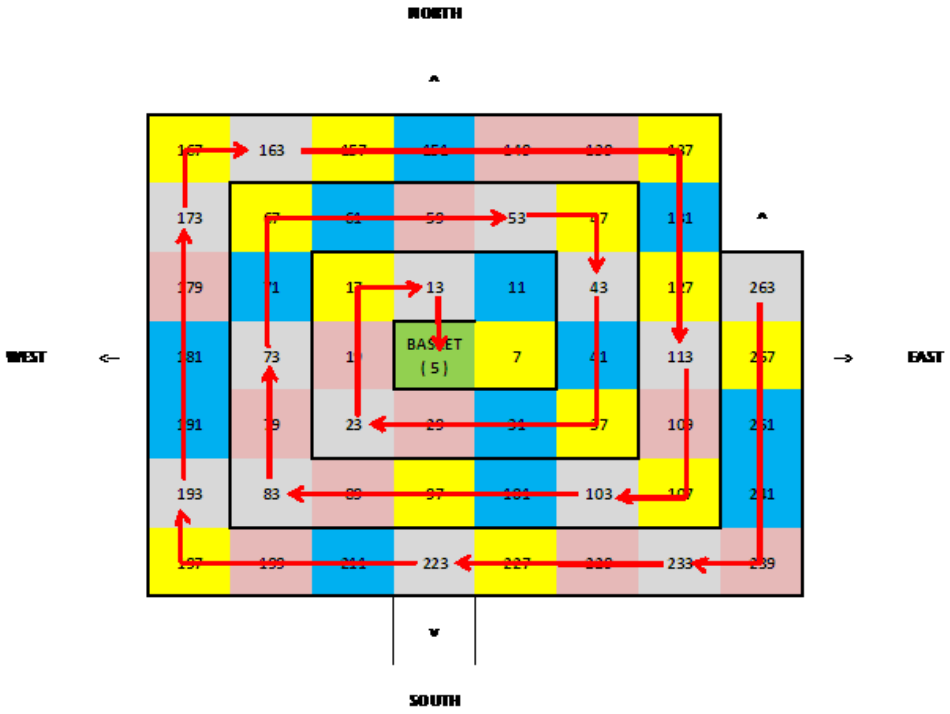
4.1 Description

The mathematics teacher in a large high school was asked to conduct some outdoor games and activities for the entire school and also divide the students into 4 different houses. He decided to check their basic math skills as well. There was a basket at the center of the playground and the place around the basket is divided into 14641 (121 * 121) "five feet squares" numbered like a typical X-Y coordinate system. He asked all students to distribute themselves equally around the basket with 5 feet distance between each other as shown in the yellow cells in the figure below. He demonstrated that with 53 students from one class. He later planned to do this for the entire school so that he can randomly create 4 house teams.



If there are additional students who cannot equally distribute themselves around the basket, they would first fill the Eastern edge starting from South (as shown in the white cells in the diagram above), and then fill the Northern edge starting from East, and then the Western Edge starting from North, and finally the Southern edge starting from West.

The students were then asked to list the prime numbers in ascending order in the form a spiral, starting with 7 as the number used by the student immediately to the East of the basket (refer figure below)



There were four large similar looking balls given to the students who were assigned the largest prime numbers ending in 1, 3,7 and 9 respectively (in this case 251, 263, 257 and 239). They had to write their number on the ball and then throw it to the next student with the same ending digit (in this case 241, 233, 227 and 229) so that they can do the same. This process had to repeat till the last student in their team (11, 13, 7, 19) who will finally throw the ball into the basket (numbered 5). The team should know who to throw to and who to catch from and any wrong or missed catches meant a penalty. The winning team is the one that has the least penalty and throws the ball into basket in the shortest time.

The sequence of throws for the team with numbers ending in 3 is depicted on the figure above.

4.2 Task

One of the sharp students decided to write a small program to help their team. Given their coordinates, it will output the prime number that he should write on the ball, the coordinate of the student from who he has to catch the ball and the coordinate of the student to whom he had to throw the ball.

You task is to help the student write the program.

4.3 Time Limit

One second for a maximum of 10000 students and 100 inputs

4.4 Input

N – number of students in the high school followed by a list of coordinates ‘xi yi’ of students who want to find out who they have to catch from and who they have to throw to. (0 < i < 100)

The end of the list is marked by '-999 -999'.

4.5 Output

For each input coordinate, the coordinates of the thrower and the catcher in the form

I should write

I should catch from

I should throw to

Where P(x,y) is the prime number to be written by the student

m, n is the coordinates of the student from whom the catch should be taken (if the input is of the first thrower – this should be recorded as 'nobody' – see example below)

p, q is the coordinates of the student / basket to whom the ball should be thrown

if the input location has no student - the the output should just have 'No student at this location' for that line

Note: Each line should end with a new line character.

4.6 Sample Input

```
53
    4 7
    4 -3
    -1 -1
    1 1
    -3 -1
    0 0
    -999 -999
```

4.7 Sample Output

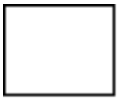
```
No student at this location
I should write 239
I should catch from nobody
I should throw to 2 -3
I should write 23
I should catch from 2 1
I should throw to 0 1
I should write 11
I should catch from 1 -1
I should throw to 0 0
I should write 191
I should catch from -1 -3
I should throw to -3 0
No student at this location
```


5 Question E: Kakuro Validation

5.1 Description

Kakuro is a number puzzle also known as “cross-sum”. The Kakuro puzzle is a n * n board which has 3 types of cells:

1. White Cells or Blank Cells in which the solver can put in any number between 1 and 9.

   represented as 0 0

2. Clue cells are Grey Cells that carry sums of sequences (row or column or both). They are not fillable but provide clues for filling values in the row and column sequences that follow.

   represented as -1 17, 14 -1 and 14, 17 respectively

3. Block Cells are Grey Cells are empty and unused.

    represented as -1 -1

The objective of the game is to fill the white cells with values that that the rules of the game are met.

The rules of the game are as follows

1. A digit cannot appear more than once in a sequence (row or column). Note that if the row or column has more than one sequence, the digit can appear once in each sequence
2. The sum of digits in a row sequence should match the row sum in the clue cell that appears ahead of the row sequence.
3. The sum of digits in a column sequence should match the column sum in the clue cell that appears ahead of the column sequence.

There are some rules that define if the structure of the Kakuro Puzzle is valid

1. All row sequences of 2 to 9 white cells should be preceded (to the left) by a Clue Cell that has a row sum
2. All column sequences of 2 to 9 white cells should be preceded (above) by a Clue Cell that has a column sum
3. No clue cells should be present where there is no row sequence or column sequence that follows it.

4. Clue Cells should have only a row sum if there is a row sequence that follows it but no column sequence below it. Similarly it should only have a column sum if there is a column sequence that follows it but no row sequence that follows it. It should have a row and column sum if it has both row and column sequences that follow.

An example of where both row and column cell are present is shown below

1. All row and column sums should have values between 3 and 45
2. The White Cells should be contiguous and connected. They should not be in the form of multiple unconnected islands. (see sample scenario 4)
3. All White Cells should have both row and column sequence neighbors. In other words, no row or column sequence can just have one member cell. See example below.

The following diagram depicts some scenarios where the structure is invalid

1. Row sum instead of column sum
2. Missing row sum
3. Orphan cell that has no column neighbor
4. Row sum with so subsequent row sequence

Additionally, there should be one and only one solution possible with the given puzzle clues.

5.2 Task

Write a program that when given a potential Kakuro Puzzle, identifies if

1. it is structurally correct
2. it has a possible valid solution
3. it has a unique solution

5.3 Time Limit

Maximum three seconds

5.4 Input

The first line of the input N is the puzzle board size (4 <= N <= 10)

This is followed by N lines of input (for each row) each containing a pair of N numbers that represent the N cells in the row.

-1 -1 represents a grey or block cell
0 0 represents a White or Blank cell that is fillable
-1 R represents a clue cell that carries only a row sequence sum (3 <= R <= 45)
C -1 represents a clue cell that carries only a column sequence sum (3 <= C <= 45)
C R represents a clue cell that carries both row and column sums

5.5 Output

One of the following

If there are structural issues – e.g. missing or extraneous row or column sums,
Invalid Puzzle Structure
If it is structurally valid and has a unique solution,
Valid Puzzle
If it is structurally valid but has no valid solution,
No Solution
If it is structurally valid but has multiple valid solutions,
No Unique Solution

5.6 Sample Input - Scenario 1

10
-1 -1 14 -1 23 -1 -1 -1 6 -1 16 -1 -1 -1 -1 -1 6 -1 6 -1
-1 17 0 0 0 0 -1 4 0 0 0 0 -1 -1 4 4 0 0 0 0
-1 11 0 0 0 0 13 11 0 0 0 0 29 8 0 0 0 0 0 0
-1 -1 -1 16 0 0 0 0 14 17 0 0 0 0 0 0 0 0 -1 -1
-1 -1 -1 -1 -1 11 0 0 0 0 0 0 0 0 9 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 7 0 0 0 0 13 8 0 0 0 0 -1 -1 -1 -1
-1 -1 -1 -1 16 -1 17 26 0 0 0 0 0 0 0 0 20 -1 -1 -1
-1 -1 4 19 0 0 0 0 0 0 0 0 11 10 0 0 0 0 5 -1
-1 19 0 0 0 0 0 0 -1 10 0 0 0 0 -1 4 0 0 0 0
-1 7 0 0 0 0 -1 -1 -1 4 0 0 0 0 -1 13 0 0 0 0

	14\	23\		6\	16\			6\	6\
\17			\4				4\4		
\11			13\11			29\8			
	\16			14\17					
		\11					9\		
		\7			13\8				
		16\	17\26					20\	
	4\19					11\10			5\
\19				\10			\4		
\7				\4			\13		

5.7 Sample Output - Scenario 1

Valid Puzzle

5.8 Explanation - Scenario 1

There is only one solution as shown below:

	14\	23\		6\	16\			6\	6\
\17	9	8	\4	1	3		4\4	3	1
\11	5	6	13\11	5	6	29\8	1	2	5
	\16	9	7	14\17	5	8	3	1	
		\11	1	3	2	5	9\		
		\7	5	2	13\8	7	1		
		16\	17\26	8	3	9	6	20\	
	4\19	3	8	1	7	11\10	2	8	5\
\19	3	7	9	\10	2	8	\4	3	1
\7	1	6		\4	1	3	\13	9	4

Note that the numbers can appear only once is a given sequence but if there are multiple sequences in each row or column, the digit can appear once in each of then e.g. 1 and 3 in the second row.

5.9 Sample Input - Scenario 2

4 -1 -1 13 -1 11 -1 9 -1 -1 5 0 0 0 0 0 0 -1 6 0 0 0 0 0 0 -1 4 0 0 0 0 -1 -1

	13\	11\	9\
\5			
\6			
\4			

5.10 Sample Output - Scenario 2

No Solution

5.11 Explanation - Scenario 2

On the second row, no 3 different digits can add to up to 5. Hence, no solution possible.

5.12 Sample Input - Scenario 3

4
-1 -1 13 -1 11 -1 9 -1
-1 23 0 0 0 0 0 0
-1 6 0 0 0 0 0 0
-1 4 0 0 0 0 -1 -1

	13\	11\	9\
\23			
\6			
\4			

5.13 Sample Output - Scenario 3

No Unique Solution

5.14 Explanation - Scenario 3

There are multiple solutions possible.
Two Examples are shown below:

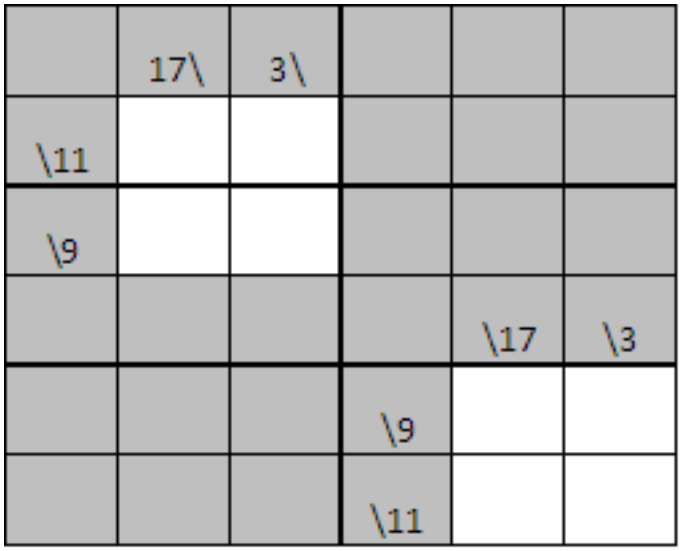
	13\	11\	9\
\23	9	6	8
\6	3	2	1
\4	1	3	

	13\	11\	9\
\23	9	8	6
\6	1	2	3
\4	3	1	

5.15 Sample Input - Scenario 4

6
-1 -1 17 -1 3 -1 -1 -1 -1 -1 -1 -1
-1 11 0 0 0 0 -1 -1 -1 -1 -1 -1

-1 9 0 0 0 0 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 17 -1 3 -1
-1 -1 -1 -1 -1 -1 -1 9 0 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 11 0 0 0 0 0



5.16 Sample Output - Scenario 4

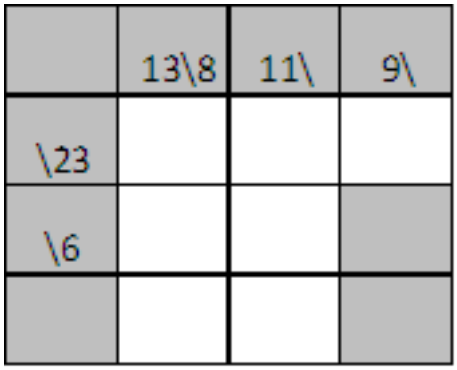
Invalid Puzzle Structure

5.17 Explanation - Scenario 4

The puzzle (though it has a valid solution) has 2 disconnected islands of White cells.

5.18 Sample Input - Scenario 5

4 -1 -1 13 8 11 -1 9 -1
-1 23 0 0 0 0 0 0
-1 6 0 0 0 0 -1 -1
-1 -1 0 0 0 0 -1 -1



5.19 Sample Output - Scenario 5

Invalid Puzzle Structure

5.20 Explanation - Scenario 5

The structure is invalid for multiple reasons

1. The last row has no row sum.
2. The last column has a white cell which is the only cell in a column sequence – this is not allowed as all white cells should have both row and column neighbors that are white cells
3. The cell with input values 13 8 is incorrect as the row sum 8 has no row sequence following it

6 Question G: Which integer is closer to “hello”?

6.1 Description

In this problem we will try to give a satisfactory answer to the question: “Which integer is closer to a predefined string?”. To be able to answer this question, we will have to develop a custom “String-Integer distance metric measure”. In order to accomplish this, we will “stand on the shoulders of giants and try to see a little further....” (Original quotation: “If I have seen further it is by standing on the shoulders of giants” Isaac Newton).

According to the number theory, each positive integer number can be written as a sum of other positive integers. This is known as the “Integer Partitioning” problem and has been studied by the best mathematical minds of all times, including the Indian genius S. Ramanujan (1887-1920). Generally, in the integer partitioning problem, two sums that differ only in the order of their summands are considered to be the same partition. Nevertheless, if the order of the summands matters then the sum becomes a composition. For example, although integer 4 has 5 distinct partitions (i.e. 4: 4, 3+1, 2+2, 2+1+1, 1+1+1+1), there exist 8 compositions which can be presented either in lexicographical or anti-lexicographical order, as presented below:

Table 1. *Compositions of the integer 4 listed in lexicographical and anti-lexicographical order.*

In Lexicographical Order	*In Anti-Lexicographical Order*
1. 1 + 1 + 1 + 1	1. 4
2. 1 + 1 + 2	2. 3 + 1
3. 1 + 2 + 1	3. 2 + 2
4. 1 + 3	4. 2 + 1 + 1
5. 2 + 1 + 1	5. 1 + 3
6. 2 + 2	6. 1 + 2 + 1
7. 3 + 1	7. 1 + 1 + 2
8. 4	8. 1 + 1 + 1 + 1

Now consider that the numbers 1-9 can be used to correspond to a set of letters as depicted at the following table:

Table 2. *Correspondence of the numbers 1-9 with a set of characters.*

1 : a, b, c	2 : d, e, f	3 : g, h, i
4 : j, k, l	5 : m, n, o	6 : p, q, r
7 : s, t, u	8 : v, w, x	9 : y, z

It now becomes clear that by replacing each summand of every composition with a letter from the set it corresponds to, we are able to form words! For example, using the 7th composition in anti-lexicographical order of the integer 4 (i.e. the composition 1 + 1 + 2), we may form the word “bad” (1→ b, 1→ a, 2→ d).

Since we have found a way to create words from integers, we only need a string similarity measure in order to decide upon the “resemblance” of the word we have formed when compared to a predefined string. Towards this direction we may employ the Dice’s coefficient metric.

According to the Dice coefficient, the similarity between two strings can be computed based on their number of matching bigrams. If the words under consideration share exactly the same bigrams, then the Dice coefficient for these strings would be equal to 1.0, whilst if they do not have any bigrams in common, the coefficient association would be equal to zero. The mathematical formula expressing the computation of the Dice coefficient between two words is shown at Eq. (1):

$$similarity_{Dice} = \frac{2 \times |bigrams(x) \cap bigrams(y)|}{|bigrams(x)| + |bigrams(y)|}$$

where bigrams is the function which reduces a word to a multi-set of character bigrams.

Consider for example the English words “transformation”, yielding the bigrams tr, ra, an, ns, sf, fo, or, rm, ma, at, ti, io, on and “transportation” yielding the bigrams tr, ra, an, ns, sp, po, or, rt, ta, at, ti, io, on. The matching bigrams between these two strings are tr, ra, an, ns, or, at, ti, io, on. It is obvious that the set of bigrams corresponding to each word consists of 13 elements, whereas only 9 of them are shared by both strings. Therefore the overall Dice coefficient association value would be $18/26 \approx 0.692$.

Based on the above, can you now develop a program that would be able to answer the question: “Which integer is closer to a predefined string?”

6.2 Task

Please write a program that would examine and write to the standard output stream the integer(s) between 1 and 10 which are “closer” to a predefined string that will be given to your program as input in run time. More specifically:

For each integer i, $1 \leq i \leq 10$, your program should be able to compute all the compositions in anti-lexicographical order of each integer i.

a. For each derived composition, all the possible strings that can be formed using the [number – character set] correspondence presented at Table 2 should be computed.

i. For each string formed from the previous step, your program should calculate the Dice coefficient value between the formed string and the predefined string that has been given to your program as input in run time.

After all the aforementioned computations have been completed, your program should print to the standard output stream the integer(s) between 1 and 10 which are “closer” to the predefined string, according to the following format:

The integers closer to the word: [predefined input word] are the following: Integer:[integer] (Word:[formed word] Dice Coeff.Value:[Dice value] from Composition:[composition])

6.3 Input

The program gets its input in the form of typical arguments in run time. It should accept only one argument that should be of type string, and it will represent the predefined word used by your program in order to compute the integers between 1 and 10 that are closer to this input word.

6.4 Output

Your program should print to the standard output stream the integer(s) between 1 and 10 which are “closer” to the word that has been given as input to your program, according to the following format:

The integers closer to the word: [predefined input word] are the following: Integer:[integer] (Word:[formed word] Dice Coeff.Value:[Dice value] from Composition:[composition]) Please note that with regards to the Dice coefficient association value, in the output it should be rounded and outputted using only the first 3 digits of accuracy. (e.g. if the Dice's value is equal to 0.4 it should be printed as 0.400. Equivalently, if it is equal to 0,6666666666 it should be outputted as 0,667).

Note: There is no new line character at the end of the result.

6.5 Sample Input 1

hello

6.6 Sample Output 1

The integers closer to the word: hello are the following: Integer:9 (Word:hel Dice Coeff.Value:0.667 from Composition:[3, 2, 4]) Integer:10 (Word:ell Dice Coeff.Value:0.667 from Composition:[2, 4, 4])

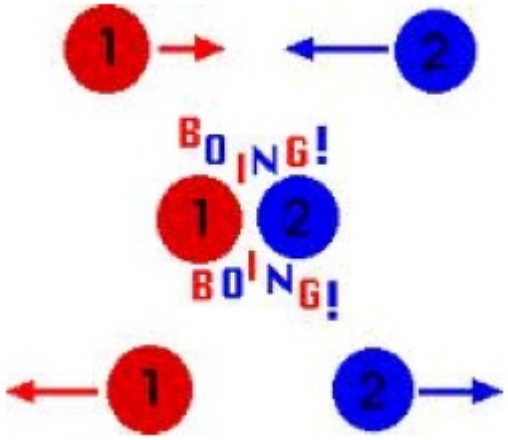
7 Question H: Tracking billiard balls

7.1 Description

A set of N billiard balls are set on a one-dimensional table. The table is 1 meter long, set north-south with two pockets at either side. Each ball has zero width and there is no friction so it is moving with a fixed velocity of either northward or southward and bounces back in a perfect elastic collision from other balls it encounter on its way (or drop into one of the pockets). Your job is to keep track of the balls movements.

7.2 Task

Please write a program that gets the initial place, speed and direction of all the balls and gives the position of a specific ball after t seconds.



7.3 Input

The first line contains the number of scenarios. Each one of the other lines in the input contains a scenario: The first number, N, is the number of balls; followed by N pairs of numbers: the distance in centimeters from the south end of the table and the speed (positive speed meaning it moves northward); the last two numbers are the number i of the target ball you should track and the time T in seconds.

7.4 Output

The output is a single number for each line which is the place (distance in centimeters from the south end of the table) of the tracked ball after T seconds. Note: There is no new line character at the end of the result.

7.5 Sample Input

```
5
1 50 1 1 1000
1 50 1 1 6
```


1 60 -2 1 6
2 10 1 95 -1 2 30
2 10 1 95 -1 2 60

7.6 Sample Output

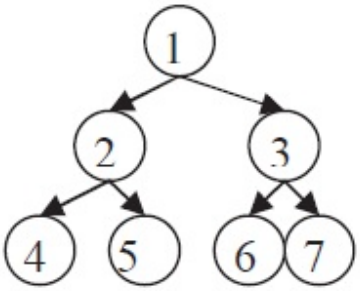
100
56
48
65
70

8 Question I: Help Bugs Bunny escape Elmer

8.1 Description

A group of Elmer Fudds are chasing Bugs Bunny in a binary tree. In every turn, Bugs moves and then all the Elmers are moving. Each move is along an edge of the tree. If any of the Elmers is on the same node as Bugs, before or after Bugs' move, they win. The tree's height is H and nodes are numbered from 1 to 2^{H-1}

For example, see tree of height 3.



Each move is one of the following four letters

U – moving up the tree (for example, from 7 to 3 in the example above)
L – moving to the left child (for example, from 1 to 2 in the example above)
R – moving to the right child (for example, from 2 to 5 in the example above)
S – stay in the same place (for example, from 4 to 4 in the example above)

All the Elmers start at the root, while Bugs can start in any node. Note that Elmer catches Bugs if he is on the same spot before or after the move; or if they cross each other on their way.

8.2 Task

Please write a program that gets the moves of the Bugs and the moves of all the Elmers and decides when he is caught or whether Bugs has escaped.

8.3 Input

The first line contains the number of problems. Each one of the other lines contains three numbers: the first is N = number of Elmers; the second is M = number of moves; and the third is P0 = Bugs's initial location. These numbers are followed by M groups of N+1 letters each which define the moves of Bugs (the first letter) and each one of the Elmers (the next N letters).

8.4 Output

The move number after which Bugs is caught, and the index of the Elmer who caught it (if several has, give the minimal one), or 0 0 if he survives the search.

Note: There is no new line character at the end of the result.

8.5 Sample Input

2
4 2 3 LLLRR SLRLR
2 6 8 LLR ULS SRS USR USU SUU

8.6 Sample Output

2 3
0 0

9 Question J: Resistors Network

9.1 Task

Can you help the puzzle-master check the solutions he receives by writing a program which, given a resistor network computes the overall resistance between "a" and "z"?

Note: we only ask you to solve networks which are parallel – series; as a hint see the clip at http://www.youtube.com/watch?v=q-DxWK5Wzzc

9.2 Input

The first line contains the number of problems. Each one of the other lines contains pairs of letters to mark the endpoint of a 1-Ohm resistor.

9.3 Output

The overall resistance, given in 4-digits accuracy.

9.4 Sample Input

4
ab bc cd dz
az az az
ab bc cz ad de ef fz
ab bc cd de ef fg gz az bg cf
Note: New line is needed at the end of the last line

9.5 Sample Output

4.0000
0.3333
1.7143
0.7321

10 Question N: Islands

10.1 Description

Vangelis the bear is the proud owner of a swamp that is composed by several small islands. He really likes his swamp but he hates swimming. In order to allow himself to enjoy his islands, he decided to build connection paths that connect those islands together.

Vangelis has at his disposal a map of the swamp. This map is a square grid with dimensions NxN (where N is greater than 1 & less than or equal to 1000) and is composed by squares that are either black or white. A black color square represents a piece of land and a white one an area of water. Two black squares are said to be connected if they are directly adjacent in the horizontal or vertical direction (not in the diagonal direction). We define an island as a set of one or more black squares such that each black square in the island is connected to at least one other black square in the island.

10.2 Task

Your task is to determine how many white squares need to be converted to black squares so that there will be only one island in the diagram with the least amount of conversions. Most test cases will have more than one solution.

10.3 Input

Your program should read the standard input. The first line of the input contains the integer N. Each of the following N lines contains N numbers on each line. The numbers on each line are either a 0 or a 1, separated by a single space. A 1 represents a black square, and a 0 represents a white square.

10.4 Output

Your program should write on the standard output a number M, the number of white squares to be converted to black.Note: There is no new line character at the end of the result.

10.5 Input 1

```
10
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```


10.6 Output 1

11

10.7 Input 2

2
1 0
0 1

10.8 Output 2

1

10.9 Input 3

4
1 0 0 0
0 0 0 0
0 0 0 0
0 0 0 1

10.10 Output 3

5

11 Question O: Patron

11.1 Description

Vangelis the bear wants to understand more about music; and giving a scientific look into songs, he realized that most songs have patterns (a.k.a motives) in them. After making this discovery, Vangelis decided to study these reappearing sequences of notes by using his computer and to do so he simplified the songs by converting them into 8bit integer values that describe only the pitch of a note. Thus, a song can be a sequence of the form 2,5,12,145,233,...

11.2 Task

Your task is, given one song encoded as above, to identify all the patterns that appear at least L times and have size of at least D notes and report back the sum of appearances of the patterns found. A pattern of size X may contain another pattern of size Y (X>Y). Patterns may overlap.

11.3 Input

Your program should read the standard input. The first line of the input contains an integer D representing the minimum allowed number of the notes within a pattern (where $1 \leq D \leq 500$). The second line contains an integer L representing the minimum allowed number of appearances of the pattern within the song (where $2 \leq L \leq 500$). The third line contains an integer F representing the number of notes within the song (where $2 \leq F \leq 10000$). Each of the next F lines contains a single integer in the range [0,255], which depicts the discrete note pitch.

11.4 Output

Your program should write on the standard output 1 line. The line should contain a single integer, in range [2,4875250], denoting the sum of appearances of the patterns that appear at least L times and contain at least D notes.

Note: There is no new line character at the end of the result.

11.5 Input

```
2
2
6
1
2
1
2
1
2
```


11.6 Output

11

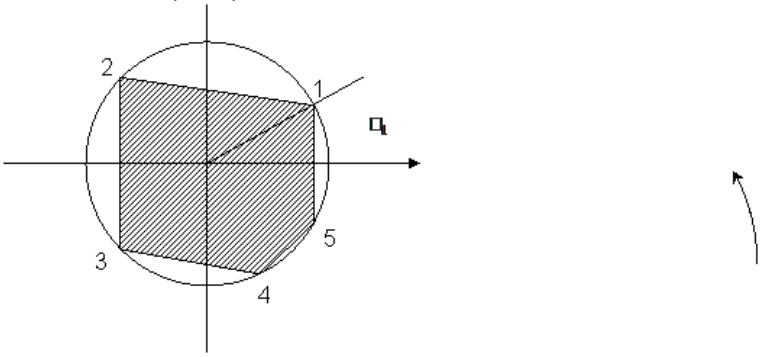
11.7 Explanation

(Pattern) - Occurrences
(1 2) - 3
(2 1) - 2
(1 2 1) - 2
(2 1 2) - 2
(1 2 1 2) - 2

12 Question S: Simple polygons

12.1 Description

Cyclic convex polygons are a simple kind of polygons: all N corners lie on a circle and the boundaries do not intersect. The surface of such a polygon is area is shown hatched in the example sketched below (N=5).



The corner positions (xi, yi) can be expressed in terms of the angle as shown in the example.

12.2 Task

Write a program that reads a set of corner point angles from stdin. The radius of the cyclic polygon is 1 (unit circle) and the angles are given as multiples of π (thus in the range 0..2). Your program calculates the surface area of the specified polygon (N<=200) and outputs it to stdout.

12.3 Input/Output format

Each line of the input file contains a corner point angle. The output is expected to have the following c-style format: %.2f.

Keep reading Input until you hit dollar sign $

Note: There is no new line character at the end of the result.

12.4 (Isosceles triangle) Input:

0
0.5
1
$

12.5 (Isosceles triangle) Expected output

1.00

13 Question V: Hogwart's Cup

13.1 Description

It's almost the end of the year; Dumbledore wants to give out the Hogwarts House cup to the group that has the most points. However recent events need to be considered and he has some last minute points to be awarded. Dumbledore could wave his wand to find the winner, but a critical bug in the wand's code made it to stop working. You need to help him, calculate the revised scores of each of the group and announce the highest score. Dumbledore is counting on you.

13.2 Input Format

First line is the number of Test Cases, N.

Next line contains Number of students X (X <= 100000), Number of Revision operations R, Number of Group values to calculate G.(#R + #G <= 100000)

Next line will be the initial scores for each of the students. Space separated s1..sX. Each student can be awarded a maximum score of 100.

Next line will continue with operations for score revision. Line starts with R, next value is the index of student to revise, and next value is the extra points awarded.

Further down will be the operations for calculating group totals. Group start student index, and end student index. (Both students inclusive)

13.3 Output Format

Highest score of the entire group score calculations

Note: There is no new line character at the end of the result.

13.4 Test Input

```
1
10 2 2
1 1 1 1 1 1 1 1 1 1
R 2 2
R 8 3
G 1 6
G 5 10
```

13.5 Test Output

```
9
```


14 Question W: Maya Diet

14.1 Description

Maya wants to switch to a healthy diet. An optimal diet consists of C gm of carbohydrates, F gm of fat, P gm of protein. Any other element in the food (represented by O) is considered unwanted. She procured a table containing the carbohydrates, fat, protein and other contents (all in gms per serving) of some healthy foods. Her goal is to consume at least C, F and P gms of carbohydrates, fat and protein respectively yet minimize the overall in-take (in gms) of food. You need to help her choose the ideal combination from the table.

14.2 Inputs

The program must read input from standard-input. The input format is as follows. The first line contains the optimal carbohydrate, fat and protein consumption (in gm), separated by spaces. The second line contains k – the number of food items. Each of the following k lines contains - name, carbohydrate, fat, protein and other contents of a food item, separated by spaces. The weight of a serving of a particular food item is the sum of C, P, F, O values on the corresponding line. The name of food items are non-space ASCII strings. The maximum value for K is 10. The maximum weight of a serving is 1000 gms. Each of the c, p, f, o values are whole numbers (integers >= 0).

C F P
k
N1 c1 f1 p1 o1
N2 c2 f2 p2 o2
N3
.
.
.
Nk

14.3 Output

Output is printed onto the standard-output. The output must contain k lines, each line containing the name of a food item followed by the number of servings Maya should consume, separated by space and in the same sequence. If no solution exists for the problem, the output must be just one word - "Null".

Note: There is no new line character at the end of the result.

14.4 Example Input 1

3 5 4
3
A 1 0 0 2
B 0 1 0 3
C 0 0 1 2

14.5 Example Output 1

A 3
B 5
C 4

14.6 Example Input 2

15 25 20
3
Banana 4 1 3 2
Orange 1 2 1 3
Apple 1 2 1 4

14.7 Example Output 2

Banana 3
Orange 11
Apple 0

15 Question X: Hacker's Challenge

15.1 Description

You are part of your national security team who is trying to breakdown on the plans of a terrorist group. Everybody has high hopes on you to obtain the magic key that will open all the enemy secrets, your other team mates have been successful in tricking the enemy to dip his hands in invisible radium ink after which he typed the secret key to unlock his precious computer. The intelligence has also found that the key is might be one among the standard words of a secret dictionary(which they have managed to sniff out from the enemies). Your counterparts have done their job, now its upto you. Being a terrific programmer, the team expects you to give them a list of probable keys which they can use to unlock the computer. So, you have the list of characters which is part of the key and the dictionary, return the list of keys as soon as possible. If you don't find any possible keys then there is a mole in your team!

15.2 Input

First line will have a number n, the number of words in dictionary. $1 <= n <= 2 * 10^5$. Followed by exactly 'n' words(separated by whitespace, each word at max having 100 charcters - only small case). Next line will have number of testcases, t. $1 <= t <= 10^3$. 't' lines follow each with a set of characters(without any spaces, max length of 26 - only small case) which forms the key.

15.3 Output

For each test case, print a line with list of all the words, separated by space, which might be probable keys in sorted order. If there is no key matching then print "NONE" (quotes are only for clarity).

15.4 Example Input

10
bill ilp lip pile pill tout out lipp cool dude
2
ilp
blyah

15.5 Example Output

ilp lip lipp pill
NONE

