# Computer Science Competition
# Region 2022
## 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 | Agustina |
| Problem 2 | Arya |
| Problem 3 | Diego |
| Problem 4 | Fai |
| Problem 5 | Ivan |
| Problem 6 | Juana |
| Problem 7 | Krithika |
| Problem 8 | Michal |
| Problem 9 | Paola |
| Problem 10 | Ricardo |
| Problem 11 | Shivam |
| Problem 12 | Tomek |

# 1. Agustina

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

Agustina and her UIL Computer Science Team are working hard this year preparing for the Regional Meet. They do daily practice tests and frequent after-school programming packets. She wants to be reminded of the six teams that won 2020-2021 UIL Computer Science Team State Championships in the six different conferences. This program will print the names of those schools.

**Input:** None

**Output:** The names of the six high schools that won the State Championship in each of the six conferences in 2020-2021. Format exactly as shown below.

**Sample input:** None

**Sample output:**

```
1A-Slidell HS
2A-San Augustine HS
3A-Fairfield HS
4A-Dallas TAG
5A-Lucas Lovejoy HS
6A-Cypress Woods HS
```

# 2. Arya

**Program Name: Arya.java** **Input File: arya.dat**

Arya is very interested in the binary search process. He wants to write a program that will allow him to see the various "guesses" the computer makes when guessing a number using the binary search.

The program will first allow for the input of N representing the largest positive integer in the possible set of numbers 1…N. Then an integer R will be input representing the target number. The program will use a binary search strategy to "find" that value.

Each line of output will display not only each of the program's guesses, but the low value and high values that were used to determine the guesses. All guesses are displayed until the target is reached. For example, if 11-21-31 is displayed, the 21 indicates the guess while the 11 and the 41 indicate the current lower and upper bounds.

If there is an odd number of items in the remaining range of numbers, the guess will be the middle number. For example, if there were 11 numbers remaining, the guess would be the 6th number in that list.

If there are an even number of items remaining, the guess will be the lesser of the two middle numbers. For example, if 20 numbers remain, the guess would be the 10th number in the list.

**Input:**
The first line of input will contain a single integer T, the number lines of data to follow ($1 \le T \le 10$).
Each line of data will consist of two positive integers N and R. N ($1 \le N \le 100000$) will represent the largest possible integer in the range. R represents the target and will be in the range ($1 \le R \le N$).

**Output:**
For each test case, sets of 3 integers will be printed on separate lines until the target is found. The first is the low value of the current range. The second is the computer's guess. The third is the high number in the current range. A dash (-) will separate the numbers with no extra spaces. After the target is found, "GOT IT!!!" will be printed.

**Sample input:**

```
5
10 7
15 12
100 32
1000 500
31 31
```

**Sample output:**

```
1-5-10
6-8-10
6-6-7
7-7-7
GOT IT!!!
1-8-15
9-12-15
GOT IT!!!
1-50-100
1-25-49
26-37-49
26-31-36
32-34-36
32-32-33
GOT IT!!!
1-500-1000
GOT IT!!!
1-16-31
17-24-31
25-28-31
29-30-31
31-31-31
GOT IT!!!
```

# 3. Diego

**Program Name: Diego.java** **Input File: diego.dat**

Diego has found a fancy digital safe but is having trouble opening it. The combination for the safe is a series of digits (0-9). Each time he tries a combination, the safe gives him two pieces of information:

1) The number of digits that belong in the correct combination from those that he picked.
2) The number of digits that are in the correct position from those that he picked.

Note that excess digits are not counted in total (1). For instance, if the code was `123`, and Diego guessed `411`, the safe's response would be `1 0` rather than `2 0`. Given a set of Diego's guesses and the safe's responses, write a program to help Diego determine how many combinations could possibly open the safe.

**Input:**
The first line of input will contain a single integer T, the number of test cases to follow ($1 <= T <= 10$)
The first line of each test case will contain two space separated integers N and M denoting the length of the safe's code ($1 <= N <= 6$) and the number of observations ($1 <= M <= 10$)
The next M lines of each test case will each consist of a single observation of the form G X Y, where G is a guess consisting of N digits (0-9), and X and Y are integers denoting how many digits belong in the code, and how many digits are in the correct position.

**Output:**
For each test case on its own line, output the number of safe combinations that are consistent with Diego's observations.

**Sample input:**
```
2
2 4
02 1 0
34 0 0
56 0 0
78 0 0
5 2
12345 0 0
67890 0 0
```

**Sample output:**
```
4
0
```

**Sample Explanation:**
Combinations that are consistent with test case 1: 10, 90, 21, 29

There are no combinations consistent with test case 2, because every digit was used and the safe reported that none of them are present in the combination

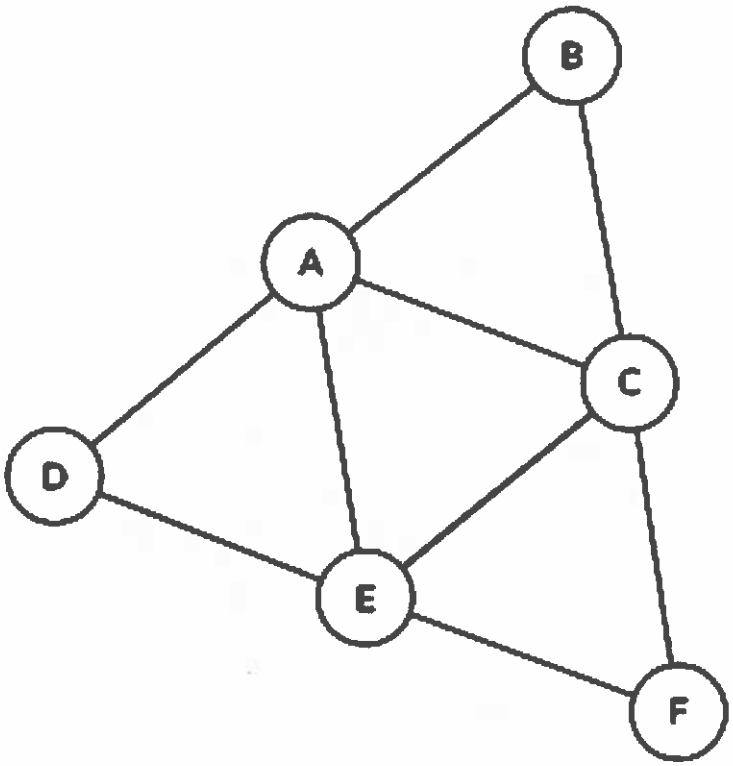
# 4. Fai

**Program Name: Fai.java** **Input File: fai.dat**

Fai just got a job working for the UIL's virtual reality team. In an attempt to get all K-12 students more familiar with their assigned district, the UIL wants to make a warped speed video showing what each student would see on a bus ride to another school's city or town. This way, students will have a general idea as to what they will see out of their school bus window before ever making the trek to the other campus.

To accomplish this, the UIL virtual reality team has provided Fai with a 360-degree camera attached to the top of her car. Fai knows she needs to drive every single road between two cities exactly one time to record the street view, so that all possible routes have been recorded. With the rising price of fuel, and in an attempt to be as efficient as Fai can be with her time, Fai knows she needs to start recording and end recording in the same city, if possible. With the task of driving every single road, Fai knows she may visit the same city or town multiple times, which is okay, she just has to travel all the given roads the UIL assigns to her to drive.

For example, say that City A, City B, City, C, City D, City E, and City F are all in the same district and the following roads exist: City A<->City B,City A<->City C,City A<->City E,City A<->City D,City B<->City C,
City C<->City F,City C<->City E,City F<->City E,City E<->City D. If Fai were to start in City A she could do the following road traversal to cover all roads exactly once, starting and stopping from City A:




City A -> City D -> City E -> City F -> City C -> City B -> City A -> City E -> City C -> City A

Can you help Fai write a program that, given a district and all the roads between cities or towns, can determine if it's possible to drive all roads starting and stopping from a given city or town?

***Continued next page…***

***Fai, continued***

**Input:** The input will consist of an integer *T*, the number of test cases. *T* will be in the range of [1,10]. For each test case, input will consist of four lines. Line 1 will contain the name of all the cities or towns in the district. The number of cities or towns will be greater than or equal to two, and will not exceed 10.. Cities or towns are not limited to one word names. It will be guaranteed that no two cities or towns have the same exact name. Names in the list will be separated by a comma ",". Line 2 contains the name of the start city or town where Fai will begin recording street views. Line 3 will consist of the roads Fai has been assigned to record by the UIL. The roads will be given in the form: "Name1<->Name2" this means a road exists between Name1 and Name2 and must be driven by Fai exactly once. A road in this problem is always two-way. Roads will be separated by a comma ",". There will be at least one road present and there will always exist a path, or series of road(s), between two cities or towns. Line 4 is 20 dashes and serves to separate all test cases.

**Output:** For each test case, you are to output "Test Case #: possible" if Fai can start and stop at the given start city, traversing each road exactly once or "Test Case#: impossible" if Fai cannot start and stop at the given start city, traversing each road exactly once.

**Sample input:**

```
9
City A,City B,City C,City D,City E,City F
City A
City A<->City B,City A<->City C,City A<->City E,City A<->City D,City B<->City C,City C<->City F,City
C<->City E,City F<->City E,City E<->City D
--------------------
Town 0,Town 1,Town 2,Town 3,Town 4
Town 3
Town 1<->Town 2,Town 1<->Town 0,Town 2<->Town 0,Town 0<->Town 4,Town 0<->Town 3,Town 3<->Town 4
--------------------
Dallas,San Antonio,Houston,El Paso,Austin
Houston
Dallas<->San Antonio,Dallas<->Houston,Dallas<->El Paso,Dallas<->Austin,San Antonio<->Houston,San
Antonio<->El Paso,San Antonio<->Austin,El Paso<->Austin
--------------------
Wall,College Station,San Angelo
Wall
Wall<->College Station,College Station<->San Angelo,San Angelo<->Wall
--------------------
Lubbock,Loredo,Lampasas,Liberty Hill,Lago Vista
Lubbock
Lubbock<->Loredo,Lubbock<->Lampasas,Lubbock<->Liberty Hill,Lubbock<->Lago Vista
--------------------
Abilene,Beaumont,Childress,Dalhart,Eden,Fort Worth
Beaumont
Abilene<->Fort Worth,Abilene<->Eden,Abilene<->Childress,Abilene<->Beaumont,Beaumont<-
>Childress,Childress<->Dalhart,Childress<->Eden,Dalhart<->Eden,Eden<->Fort Worth
--------------------
Alice,Big Spring,Colorado City,Denton,Eagle Pass,Frenship,Goliad
Frenship
Alice<->Big Spring,Alice<->Colorado City,Big Spring<->Colorado City,Big Spring<->Denton,Big Spring<-
>Eagle Pass,Colorado City<->Denton,Colorado City<->Frenship,Denton<->Eagle Pass,Denton<-
>Frenship,Frenship<->Eagle Pass,Frenship<->Goliad,Eagle Pass<->Goliad
--------------------
Caldwell,Bryan,Navasota,Hempstead,Franklin,Calvert
Bryan
Caldwell<->Bryan,Bryan<->Navasota,Navasota<->Hempstead,Hempstead<->Franklin,Franklin<-
>Calvert,Calvert<->Hempstead,Bryan<->Calvert
--------------------
Texline,Brownsville
Texline
Texline<->Brownsville
--------------------
```

**Sample output:**

```
Test case 1: possible
Test case 2: possible
Test case 3: impossible
Test case 4: possible
Test case 5: impossible
Test case 6: possible
Test case 7: possible
Test case 8: impossible
Test case 9: impossible
```

# 5. Ivan

**Program Name: Ivan.java** **Input File: ivan.dat**

Ivan has worked on an algorithm for scoring UIL written exams. All questions will be multiple choice with 5 choices, only one that is correct. Correct answers will earn 6 points while incorrect answers will be penalized 2 points. Questions that are unanswered earn no points. He is unsure of his string handling skills and needs you to validate his algorithm.

Can you help Ivan implement his scoring algorithm?

**Input:** First line of data file contains a positive integer T, the number of exams that will be scored with $1 \leq T \leq 25$. The next line will contain a single string with exactly 40 uppercase letters from { A, B, C, D, E }. Each letter is the correct answer for one of the 40 questions with the first letter for the first question and the remaining letters in sequence to the last question. The following T lines will then contain a single string with exactly 40 uppercase letters from the same set or an underscore '_' which indicates the question was not answered.

**Output:** For each exam, display one line with the score and the percentage of attempted questions that were correct. Format the line as shown below with the percentage correct rounded to 1 decimal place. If no questions are attempted, set the percentage to 0.0.

**Sample input:**

```
4
BBEEBCECDCCCCDDBDAEBBBECAEBCBBECAEBCDEED
BBEEBCECDCCCCDDBDAEBBBECAEBCBBECAEBCDEED
BCEEBAECDCDCCDDADAEBEB_CAEBCB_ECAE_CD___
AB_DEA_CDEAB_DEABCD_ABCDEAB_DEABC_EABCD_
BACBEDACBACBEDAAECBDBADCDBBAACCDABECBAAD
```

**Sample output:**

```
Exam #1: 240 100.0
Exam #2: 164 85.3
Exam #3: -18 18.2
Exam #4: 0 25.0
```

```
int l: parse Int(nextLine)
String key: nextLine
for(){ String l: Line
  int ath: 0
  int w: 0
  int r: 0
  for(int i:0; i<40; i++){
    if(l.charAt(i):'_')
      continue
    if(l.charAt(i):key.charAt(i)){
      ath++
    }
    else
      w++
  int s: 6*w - 2*r
  Double acc: ath/40*100.0
  printf("Exam #%d: %d %6.1f", s, acc)
```

***Juana, continued***

**Sample input:**

```
3
4 5
    1     2     3     4     5
    6     7     8     9    10
   11    12    13    14    15
   16    17    18    19    20
3 2
7 6
    1     2     3     4     5     6
    7     8     9    10    11    12
   13    14    15    16    17    18
   19    20    21    22    23    24
   25    26    27    28    29    30
   31    32    33    34    35    36
   37    38    39    40    41    42
5 4 2 3
3 3
    1     2     3
    4     5     6
    7     8     8
2 2 3 1
```

**Sample output:**

```
Test case #1:
   12   13   14
   17   18   19
--------------------
Test case #2:
   28   29
   34   35
   40   41
--------------------
Test case #3: Unable to extract requested size!
--------------------
```

# 6. Juana

**Program Name: Juana.java** **Input File: juana.dat**

Juana has been working with 2-dimension tables of data and would like to extract an arbitrary "chunk" of the data. She will provide the number of rows and columns in the original table along with the data to populate the table. She will then provide a starting point by identifying the row and column positions of the top left corner of the "chunk" along with the numbers of columns and rows desired. Juana does not have programming experience so her tables start with row 1 and column 1 at the top-left corner. She has provided the following example:

| | Col 1 | Col 2 | Col 3 | Col 4 | Col 5 |
|---|---|---|---|---|---|
| Row 1 | 1 | 2 | 3 | 4 | 5 |
| Row 2 | 6 | 7 | 8 | 9 | 10 |
| Row 3 | 11 | 12 | 13 | 14 | 15 |
| Row 4 | 16 | 17 | 18 | 19 | 20 |

The table contains 4 rows and 5 columns with the data as shown above. Juana would like to extract the "chunk" of data that starts at row 3 and column 2. She wants 3 columns across the rows and 2 total rows as shown by the shading above.

Can you create a "chunk" extraction program for Juana?

**Input:** First line of data file contains a positive integer T, the number of test cases that follow with $1 \leq T \leq 10$. Each test case starts with a line containing the number of rows R and columns C for the table with $2 \leq R,C \leq 15$. That line will then be followed by R lines of data each containing C integers in the range [0, 1000). The data items will be right-aligned with leading spaces for student viewing below. The next line will contain 4 integers separated by a space. The first pair of integers are the row and column numbers of the top-left corner. The next 2 integers are the count of columns and rows desired in the "chunk". All 4 integers will have the same range as specified for R and C above. There is no guarantee that Juana specifies the starting point and sizes for the "chunk" correctly. When there is not enough data to extract the complete "chunk" an error message will be displayed instead of the requested "chunk".

**Output:** For each test case, display a line with the test case number formatted as shown in the sample. If the "chunk" cannot be extracted, display the error message as shown below on the same line as the test case. Otherwise, the extracted "chunk" of data is displayed below the test case line. Each data item is right-aligned in a column that is 5 positions wide. Follow the "chunk" with a line containing 20 hyphens "--------------------".

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

# 7. Krithika

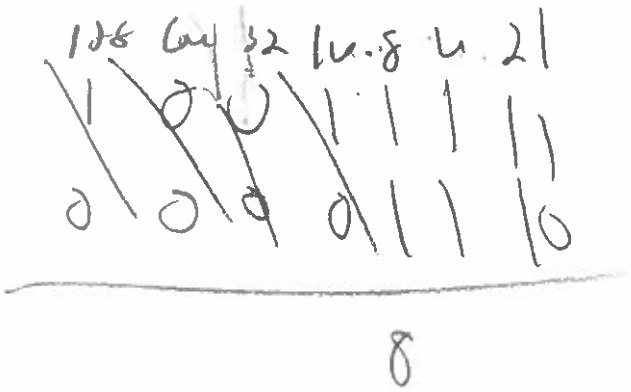
**Program Name: Krithika.java** **Input File: krithika.dat**

You are given an array A of length N. An integer X is a k-"array factor" of A if the bitwise AND of some k elements (not necessarily consecutive) of A is equal to X. Given A and k, find the largest k-"array factor" of A.

**Input:**
The first line of input is T (1 <= T <= 50), the number of test cases. The first line of each test case has space-separated integers N and k, where N (1 <= N <= 1,000) is the number of elements in the array and k (1 <= k <= N) is the sought k-array factor. The second line of each test case contains N positive integers in the range [1, 10^18], the elements of A. Note that the elements of A may not fit into a 32-bit integer data type.

**Output:**
For each test case, output the largest k-"array factor" of A. Format the output with the case numbers as in the samples.

**Sample input:**
```
3
3 2
14 159 26
4 3
1 1 1 2
5 4
1 2 3 4 5
```



**Sample output:**
```
Case #1: 26
Case #2: 1
Case #3: 0
```

**Sample Explanation:**
The 2-array factors of the first array are 14, 10, and 26. Of these, 26 is the largest.

# 8. Michal

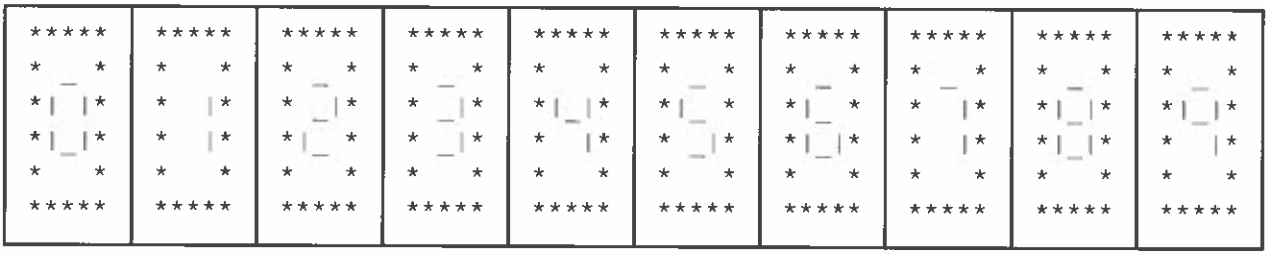
**Program Name: Michal.java** **Input File: michal.dat**

Michal's school just got a new scrolling marquee sign, but the problem is, the sign didn't come with a controller! Michal's principal has asked Michal to write a program to convert all input into text to be displayed on the sign. Michal knows this is a big undertaking, so he has decided to start with only the numerical digits 0-9 first.

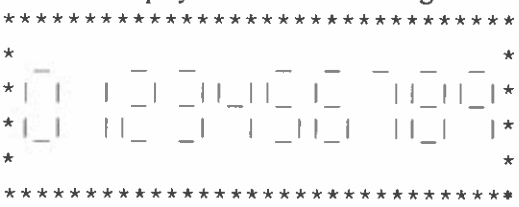
The sign utilizes a seven-segment display. For example, if all 7 segments are turned on, you would get the numeral 8 as shown below:

```
*****
*   *
* _ *
*|_|*
*|_|*
*   *
*****
```

With a combination of turning certain segments on or off, all digits 0-9 can be achieved as the table below shows:



Outputting one number at a time wouldn't be too hard programmatically, a switch statement or if-else statement could get the job done in a jiffy. The problem is, Michal's principal wants to utilize the sign's full potential and display multiple numbers at one time. For example, if Michal's principal wanted to display "0123456789" the sign would need to display:



Given an input number, can you help Michal write a program that outputs the input number on a single line surrounded by a box of asterisks ('*')?

**Input:** The input will consist of an integer *T*, the number of test cases. *T* will be in the range of *[1,20]*. For each test case, input will consist of a single number with a minimum of one digit, and a maximum of 20 digits. The input will have no spaces and will consist of the digits [0-9] only.

**Output:** For each test case, you are to output the number using the seven segment table above, in a single bounded box of asterisks

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

*Michal, continued*

**Sample input:**

```
7
8
0123456789
9876543210
2022
01010001
246810
13579
```

**Sample output:**

```
*****
*   *
*|_|*
*|_|*
*   *
*****
*********************************
*  _     _  _     _  _  _  _  _  *
*| |  | _| _||_||_ |_   ||_||_|*
*|_|  ||_  _|  | _||_|  ||_|  |*
*                               *
*********************************
*********************************
*  _  _  _  _  _     _  _     _  *
*|_||_|  ||_ |_ |_| _| _|  ||  |*
*  ||_|  ||_| _|  | _||_   ||_|*
*                               *
*********************************
**************
*  _  _  _  _ *
* _|| | _| _|*
*|_ |_||_ |_ *
*           *
**************
**************************
*  _     _     _  _  _    *
*| |  ||  |  || || || |  |*
*|_|  ||_|  ||_||_||_|  |*
*                        *
**************************
********************
*  _     _  _     _  *
* _||_||_ |_|  ||  |*
*|_   ||_||_|  ||_|*
*                  *
********************
****************
*     _  _  _  _ *
*  | _||_   ||_|*
*  | _| _|  |  |*
*             *
****************
```

# 9. Paola

**Program Name: Paola.java** **Input File: paola.dat**

Paola absolutely loves isosceles right triangles. She also is intrigued by the alphabet. This program will combine those two things. You will create a program that creates a right triangle using a continuous string of letters. The program will accept the input of an uppercase letter representing the starting letter. It will also accept the input of the number of rows to be created. The program will then display a right triangle of the shape shown below - with a hypotenuse having a positive slope.

```
     *
    **
   ***
  ****
 *****
******
```

Instead of asterisks, the program will output letters of the alphabet beginning with the input character. The letters will proceed in order as shown below. If the letter 'Z' is reached, the next letter will be an 'A'. In the example below, the inputs were M and 6.

```
     M
    NO
   PQR
  STUV
 WXYZA
BCDEFG
```

**Input:**
The first line of input will contain a single integer T, the number data lines to follow (1 <= T <= 10).
Each line of data will consist of an uppercase letter Ch ('A' <= Ch <= 'Z'), and an integer N (1 <= N <= 20) representing the number of rows that the triangle will use.

**Output:**
For each test a triangle will be produced.

**Sample input:**

```
5
A 3
Z 5
D 7
M 1
J 10
```

**Sample output:**

```
  A
 BC
DEF

    Z
   AB
  CDE
 FGHI
JKLMN

      D
     EF
    GHI
   JKLM
  NOPQR
 STUVWX
YZABCDE
M

          J
         KL
        MNO
       PQRS
      TUVWX
     YZABCD
    EFGHIJK
   LMNOPQRS
  TUVWXYZAB
CDEFGHIJKL
```

# 10. Ricardo

**Program Name: Ricardo.java** **Input File: ricardo.dat**

Ricardo has been following his New Years resolution of regularly going to the gym. At this point in time, he has gotten comfortable with many different exercises. In fact, he is comfortable with so many exercises, he has decision paralysis and cannot decide which exercises to do when he goes to the gym.

To help with this, Ricardo has developed "workout plans". Each workout plan is one of the following:

1. A single exercise (e.g. "`squats`"). All exercise names are made of lowercase English letters.
2. An option between multiple workout plans. These are surrounded by parentheses and separated by '`|`' characters. (e.g. "`(rows|curls)`"). This means that Ricardo can choose to either do rows or curls.
3. A sequence of workout plans in order. These are surrounded by parentheses and separated by '`,`' characters. (e.g. "`(running,deadlift,stretch)`"). This means that Ricardo runs, then does deadlifts, then stretches.

Since workout plans can nest, these plans can get quite complicated, and there can be many options. Now, Ricardo is wondering, how many different workouts can he complete given a plan? Two plans are different if Ricardo makes a different decision when presented with an option. Note that different exercises can have the same label. For example, there are two different workouts for the plan "`(a|a)`".

**Input:**
The first line of input is T ($1 \le T \le 30$), the number of test cases. Each test case is a single workout plan. Each workout plan consists only of lowercase letters, parentheses, and the '`|`' and '`,`' characters. No workout plan has more than 200 characters.

**Output:**
For each test case, output the number of workout plans that Ricardo can do. Format your answer with the case number as in the samples. It can be proven that given the bounds on the input data, the total number of workouts will fit into a signed 64-bit integer data type.

**Sample input:**

```
4
(rows|curls)
(running,deadlift,stretch)
(lunge,(rows|curls),(squats|press))
(a|a)
```

**Sample output:**

```
Case #1: 2
Case #2: 1
Case #3: 4
Case #4: 2
```

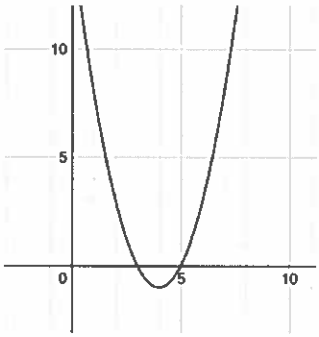
**Sample Explanation:**
In the third test case, these are the possible workouts:

1. lunge, rows, squats
2. lunge, rows, press
3. lunge, curls, squats
4. lunge, curls, press

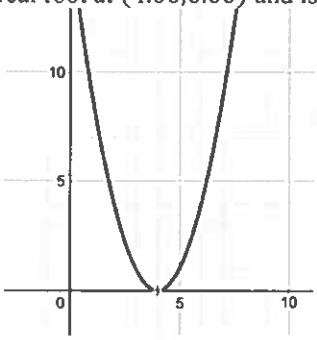
# 11. Shivam

**Program Name: Shivam.java** **Input File: shivam.dat**

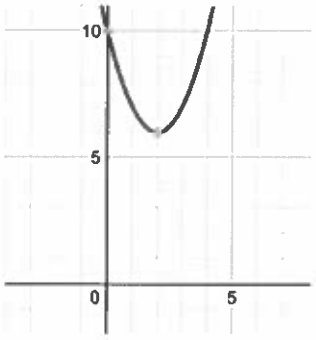
In Shivam's Algebra I class, Shivam's teacher just taught his class about quadratic functions. A quadratic function is any function that can be written in the form: $f(x) = ax^2 + bx + c$ where $x$ represents an unknown variable, the coefficients of the function are $a$, b, and $c$, and $a \neq 0$. When graphed on the $xy$ plane, quadratics are known for their "U" shaped appearance. For example, the function $f(x) = x^2 - 8x + 15$ is graphed below:




Where the function intersects the $x$ axis, of the $xy$ plane, is known as the function's root(s). In the above example, the function has two real roots, one at $(3.00,0.00)$ and another at $(5.00,0.00)$. Not all quadratics have two real roots, though. For example, the function $f(x) = x^2 - 8x + 16$ only has one real root at $(4.00,0.00)$ and is graphed below:




Some quadratics have no real roots meaning their graph does not intersect the $x$ axis at all. For example, the function $f(x) = x^2 - 4x + 10$, which is graphed below, shows an example of a quadratic that doesn't intersect the $x$ axis at all.




Shivam needs your help writing a program that can read in a quadratic function $f(x)$, determine the number of roots, and where those roots are. Can you help him with this?

***Continued next page…***

*Shivam, continued*

**Input:** The input will consist of an integer $F$, the number of functions. $F$ will be in the range of $[1,20]$. The following $F$ lines will each contain a single function $f(x)$ of the form `f(x)=ax^2+bx+c`. There will be no spaces in the function input. For this program, the caret operator (`^`) will be used for exponents and not the `xor` operator. `a` will be in range of $[-100,0) \cup (0,100]$, `b` and `c` will be in range $[-100,100]$. $a$, $b$, and $c$ are all guaranteed to be integers, but $b$ and/or $c$ are not guaranteed to be present in the function input. For example, the function `f(x)=4x^2+8` is a legal input in which only coefficients $a$ and $c$ are present.

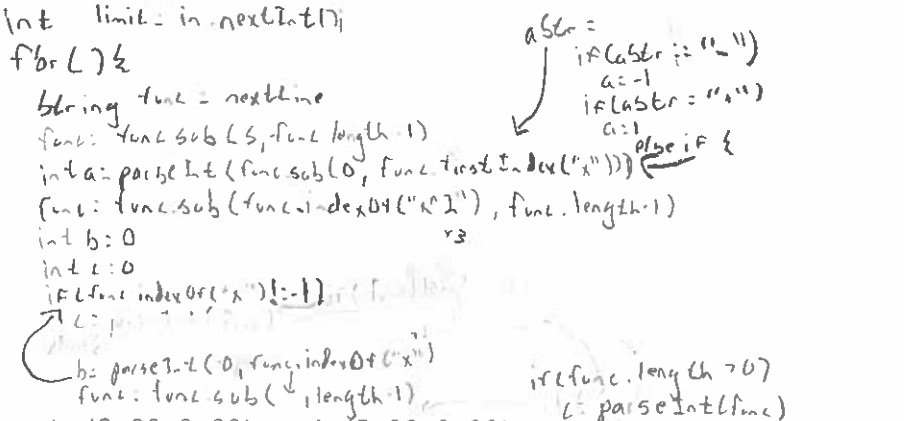
**Output:** For functions with two real roots, you are to output "`Function #: There are two real roots at (ROOT1_X,ROOT1_Y) and (ROOT2_X,ROOT2_Y).`" Roots should be displayed in ascending order according to the x component and rounded to two decimal places. For functions with one real root, you are to output "`Function #: There is one real root at (ROOT1_X,ROOT1_Y).`" The root should be rounded to two decimal places. For functions with no real roots, you are to output "`Function #: There are no real roots to the function.`"

**Sample input:**

```
9
f(x)=x^2-8x+15
f(x)=x^2-8x+16
f(x)=x^2-4x+10
f(x)=-23x^2-25x
f(x)=4x^2+8
f(x)=-78x^2+32x+6
f(x)=-89x^2+6
f(x)=3x^2+54
f(x)=x^2+10x+0
```

**Sample output:**

```
Function 1: There are two real roots at (3.00,0.00) and (5.00,0.00).
Function 2: There is one real root at (4.00,0.00).
Function 3: There are no real roots to the function.
Function 4: There are two real roots at (-1.09,0.00) and (0.00,0.00).
Function 5: There are no real roots to the function.
Function 6: There are two real roots at (-0.14,0.00) and (0.55,0.00).
Function 7: There are two real roots at (-0.26,0.00) and (0.26,0.00).
Function 8: There are no real roots to the function.
Function 9: There is one real root at (0.00,0.00).
```



# 12. Tomek

**Program Name: Tomek.java** **Input File: tomek.dat**

The search for new habitable land is on! The Universe's Inhabitation Legion (UIL) is on the hunt for new places for the galaxy's displaced to live. Currently, the UIL is investigating the distant planet Xac-12, which was thought to be inhospitable to humans.

On Xac-12, humans cannot survive on the mainland because of human-hunting predators. However, the UIL has found a large lake. Inside the lake are islands which are devoid of any predators. These islands warrant further investigation, but the UIL only has grainy images to work with. Each image is an R by C grid of tiles, and each tile is either water or land. Two tiles are connected if they share an edge. In the photos the UIL has taken, there is at most one connected body of water (the lake), and potentially many connected bodies of land. A connected region of land is an island if it is fully within the lake. Any land on the border of the image can be assumed to be mainland, and not an island.

Can you write a program to help the UIL find all islands in the lake, and their sizes? The size of an island is the number of cells it takes up in the image. Output the sizes in descending order.

**Input:**
The first line of input is T (1 <= T <= 20), the number of test cases. Each test case starts with two integers R and C (3 <= R, C <= 100), the number of rows and columns in the input image respectively. Then follow R lines with C characters each. All characters are either `'.'`, signifying land, or `'W'`, signifying water. In each test case, there is at most one connected body of water.

**Output:**
For each test case, output two lines. On the first line, output the case number and the total number of islands, formatted as in the samples. On the second line, output a space separated list of island sizes. Output the island sizes in descending order. If there are no islands in the image, output the string "NONE".

**Sample input:**

```
3
5 5
WW...
WWW..
W.WW.
W.WW.
WWWWW
3 3
...
.W.
...
4 6
.WWWW.
W..W.W
WW.WWW
.WWWW.
```

**Sample output:**

```
Case #1: 1
2
Case #2: 0
NONE
Case #3: 2
3 1
```

**Sample Explanation:**
In the first test case, there are two masses of land. The landmass in the upper right is part of the mainland, so there is only one island of size 2.

In the second test case, there are no islands.

In the third test case, there are two islands. The land in each corner of the image is part of the mainland.