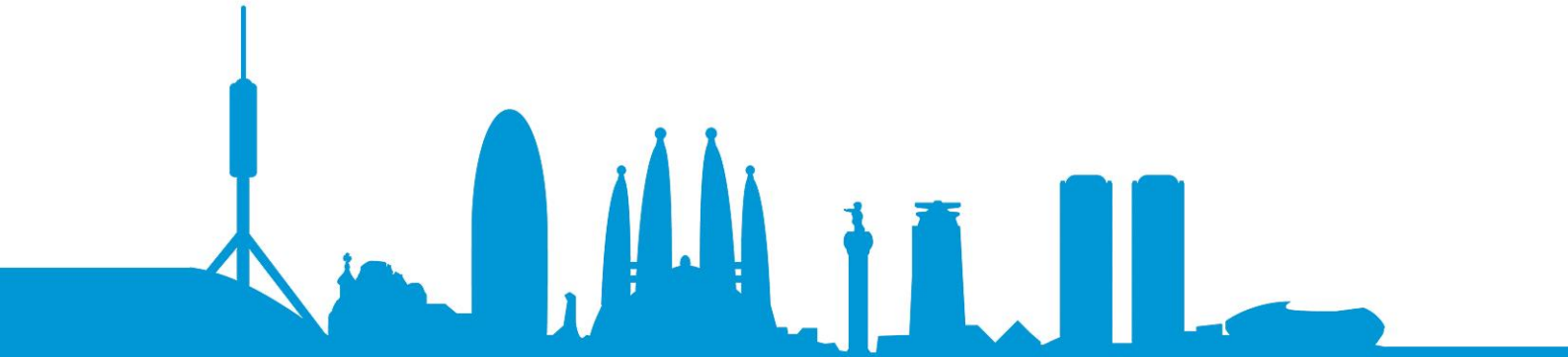


CodeWars 2015
Barcelona

Problems with solutions

Important contest instructions

Please, read the following instructions carefully. They contain information on how to package and submit your solutions to the judges. If you have any questions regarding these instructions, please ask a volunteer before the start of the contest.

Program input/output

Most of the problems (except problem 9) will accept input directly from the keyboard and will output directly to the screen instead of performing file operations.
Some programs may have inputs that are too long to be written with a keyboard. Fortunately, operating systems provide a mechanism that allows to simulate the keyboard using a text file. The program will believe to be reading from the keyboard but, without the need to perform any file operation, a file will be used. This is called a redirection and can be done with the command line. The syntax for this is to include a '<' and the file that the operating system will emulate, as it was typed in the keyboard. For example:

```
%> java prob04 < prob04.txt
%> python prob04 < prob04.txt
%> prob04.exe < prob04.txt
```

In these examples, you are executing prob04 in java, python or directly a compiled one and you are, for all of them, asking the operating system to emulate a user typing the contents of the file prob04.txt into the keyboard. This way, your programs should behave exactly as it would if you were typing the contents of the file with your keyboard.

Programming languages and what to submit

The following languages are admitted: C, C++, Java, Javascript and Python (either 2.7 or 3.x).
All the programs must be named probXX where XX is the number of the program, from 00 to 24.
If you are using a fully interpreted language, like Python or Javascript, you have to submit the source file, .py2 (for python 2.x), .py3 (for python 3) or .js.
If you are using Java, you have to commit the .java file. The main class must be named probXX. All main and supporting classes must be in the default (or anonymous) package.
If you are using a compiled language, C or C++, you have to commit the Windows executable file (see Operating system below).

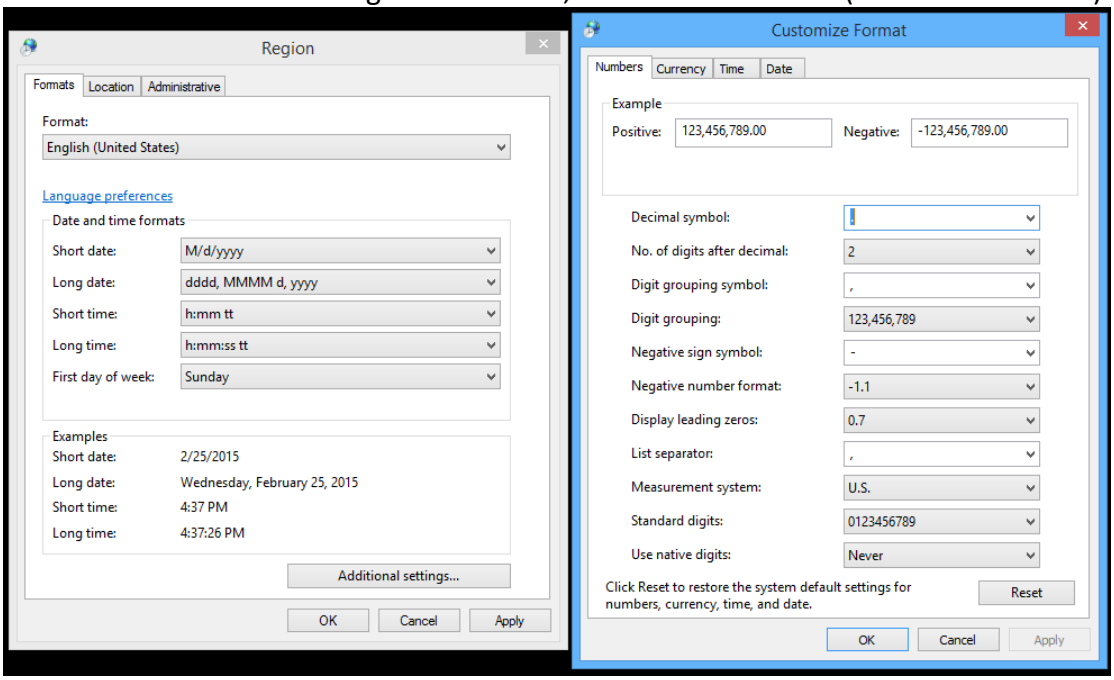
Operating system

Please be aware that our judges are using Windows to correct your problems so, if your program is compiled, you must produce a Windows executable file. If you use Linux or MacOS and want to use a compiled language (C, C++), you have to make sure your compiler is producing a windows .exe file.
This does not happen with interpreted languages (Python, Java, Javascript), since these files can be executed in Windows, Linux or MacOS.

English number notation

In English, comma (,) is used for thousand grouping, while dot (.) is used for decimals. For instance: 1,678.354 is read "one thousand six hundred and seventy-eight and 354 thousandths."

All the problems use this notation but you can find that your computer is using the Spanish configuration. Please be aware that, depending on your environment and the language or SDK you use, your program can be configured to understand US English or Spanish notation.
You can control these settings on Windows, in the Control Panel ("Panel de control") and Region



Precision when using float/double/….

It is recommended to output only 2 decimals (either rounded or truncated) when the output of the program is a number with decimals (types float, double….) but this is not a must and it is not compulsory to do any change.

Useful definitions

The following definitions can be useful for your problems.

Divisible/Remainder/Modulo operation

Divisible numbers

Two numbers are divisible when the reminder of their natural division equals 0. For example, 8 is divisible by 4, 15 is divisible by 3, 100 is divisible by 10...

Remainder

The remainder of a division is the integer "left over" after dividing an integer by another.
Examples:

- If you divide 9 by 2, the result is 4 and the remainder is 1.
- If you divide 17 by 3, the result is 5 and the remainder is 2.

Modulo operator

It is the operator that outputs the remainder of the division. It is expressed by % in all the supported languages. For example:

`17 % 3` would be evaluated as `2`

Leap year calculation

A leap year is that one with 366 days, when February has a 29th day.
This happens when the year number is multiple of 4 (divisible by 4) except for the multiples of 100 that are not multiples of 400. For example, 2000 is leap but 1700, 1800 and 1900 are not.

Specific instructions per language

C/C++

Please remember that you have to create the `main` function. This is the main code of your program.

```
int main (int argc, char** argv)
```

Java

Please remember that the main class must be named as the problem (for example, `prob00`) and must be outside any package.
Some IDE's like NetBeans create a package when you create a project. This is needed to compile with the default settings. But, when you are ready to submit, remember to delete that line so that the program runs in the jury's computer!

For example:
`package prob00;` ← this line must be removed when you ready to submit

Remember also that your class must contain the main function:

```
public static void main(String[] args)
```

Python

Since we are using two different versions of Python that use different syntax, please use the appropriate extension for your problems, depending on the version you use. Do not use `.py`

If you are using Python 2.x, your problem must be `.py2`, for example `prob00.py2`
If you are using Python 3, your problem must be `.py3`, for example `prob00.py3`

Remember that this is the 1st HP CodeWars in Europe and that you are making history, do your best to solve the problems, don't give up but if you get stuck in a problem, go to the following! We wish you the best of the lucks!

0 We are at HP!

1 point

NOTE: This is the first of two problems that can be solved and submitted before the start of the CodeWars competition. Teams are **strongly** encouraged to submit these problems **prior** to the start of the competition – hey, it's basically a free point!

Introduction

Write a program that writes the welcome message, including your group number.

Input

There is no input for this program.

Output

The program must output the following text, indicating your group number:

```
Hello world, we are at HP. We are group number 3.
```

Solution

```
public class Prob00{

    public static void main(String[] args) {

        int num;
        Scanner leer=new Scanner(System.in);
        //input data
        num=3;
       //output
        System.out.println("Hello world, we are at HP. We are group number " +
num + ".");
    }
```

1 Let's begin introducing ourselves

1 point

NOTE: This is the second of two problems that can be solved and submitted before the start of the CodeWars competition. Teams are **strongly** encouraged to submit these problems **prior** to the start of the competition – hey, it's basically a free point!

Introduction

You'll have no chance to win at HP CodeWars (or life) if you don't know how to do Input and Output properly. You also won't do well at CodeWars if you are rude to your judges. Write a program to greet your esteemed judges appropriately. Read in the name of a judge and output your greeting in the appropriate format. If you're confused at this point, go back and re-read your contest instructions.

Input

The input will be your judge's first name, a single word with no spaces:

```
Simon
```

Output

Welcome your judge with a friendly, creative greeting of some sort that includes the judge's name (does not have to match the below example):

```
Greetings, Simon the Great! I genuflect in your general direction!
```

Solution

```
public class Prob01 {
    public static void main(String[] args) {

        Scanner read=new Scanner(System.in);
        // input data
        String judge_name;
        judge_name=read.next();
       //output
        System.out.println("Greetings, "+ judge_name+ " the Great! I genuflect
in your general direction!");
    }

}
```

2 Trip to England

2 points

Introduction

You are driving your car to England (yes, you can cross the Channel with a ferry) and you have learnt that International Metric System is not used there. Like other former colonies, they use the Imperial System that uses miles to measure distances between cities.
You need to write a program that converts miles into kilometers. A mile is 1.609 Km.

Input

The input will be a distance, expressed as a positive float number.

```
2.57
```

Output

The program must output the list of distances in miles with their conversion to Kilometers.

```
2.57 miles are 4.14 kilometers
```

Solution

```
public class Prob02{

    public static void main(String[] args) {
        double millas, km;
        Scanner leer=new Scanner(System.in);
        millas=leer.nextFloat(); //read a number
        leer.nextLine();
        km =  millas * 1.609;
        System.out.println(millas + " miles are " + km +" kilometers" );
    }

}
```

3 How old is my neighbor's Chihuahua?

2 points

Introduction

Usually pet owners tend to wonder what would be the "human age" of their pets. In my case I wonder what is the "human age" of my noisy neighbor's Chihuahua.
Experts consider that each year of the Chihuahua development is the equivalent of 10 human years on the first two years of life, while for the rest of their lives a year would be equivalent to 4 years of a human being.
For example, for a 5 year old Chihuahua, its human age would be 2*10 + 3*4 = 32 years old.

Input

The first line of the input will be the Chihuahua's age in years (always less than 256). The second line of the input is its name:

```
3
Rex
```

Output

The output should be the Chihuahua's human age in the form:

```
Rex is 24 human years old
```

Solution

```
import java.util.Scanner;

public class Prob03 {

    public static void main(String[] args) {

        int dog_age, human_age;
        String dog_name;
        Scanner leer=new Scanner(System.in);

        //input
        dog_age=leer.nextInt(); //read a number (dogÂ´s age)
        leer.nextLine();
        dog_name=leer.next(); //read a name (dogÂ´s name)

        //operations
        if (dog_age>2)
        {
            human_age = 20 + (dog_age - 2) * 4;
        }
        else
        {
            human_age = dog_age * 10;
        }

        System.out.println(dog_name+" is " + human_age+" human years old" );
    }
}
```

4 Prime numbers

3 points

Introduction

A prime number is an integer that is only divisible by itself and 1.
You have to write a program that determines whether a number is prime.

WARNING: You cannot use language-provided functions

Input

The input of the program is a set of numbers, ending with a zero.

```
7
26
177
509
2053
0
```

Output

The program must output whether each number is prime.

```
7 is prime
26 is not prime
177 is not prime
509 is prime
2053 is prime
```

Solution

```
import java.util.Scanner;
import java.io.BufferedReader;


public class Prob04 {

    public static void main(String[] args) {
        int num;
        Scanner read = new Scanner(System.in);
        num = read.nextInt(); //read a number

        while (num != 0){
               boolean isPrime = num <= 3 || num % 2 != 0; //The numbers from 1
to 3 are prime and all pair numbers are not prime (except 2)
                int divisor = 3;
                while (divisor * divisor <= num && isPrime) //The highest
divisor is the sqr root of the number
                {
                      if (num % divisor == 0) isPrime = false;
                      else divisor += 2; //Skip pair numbers
                }
            if (isPrime) System.out.println(num + " is prime ");
            else System.out.println(num + " is not prime ");

                num = read.nextInt(); //read next number
        }
    }
}
```

5 Draw, o, coward!

4 points

Introduction

A palindrome is a phrase that reads the same forward or reversed.
You should write a program that is able to determine whether a phrase is a palindrome or not, skipping all punctuation symbols and not distinguishing between upper and lower case.

Input

The input will be a sequence of sentences, ending with the sentence “Palindrome!”.

```
As I pee, sir, I see Pisa
I am writing a palindrome program
Mr owl ate my metal worm
Palindrome!
```

Output

The program must determine if the phrases are palindromes (be careful with capital letters, spaces, commas and dots!).

```
“As I pee, sir, I see Pisa” is a palindrome
“I am writing a palindrome program” is not a palindrome
“Mr owl ate my metal worm” is a palindrome
```

Solution

```
import java.util.Scanner;

public class Prob05 {

	public static void main(String[] args) {

		String text;
		Scanner read = new Scanner(System.in);
		text = read.nextLine();
		while (!text.contentEquals("Palindrome!"))
		{
			String cleanText = "";
			String invertedText = "";
			for (int x=0; x < text.length(); x++)
			{
				char c = text.charAt(x);
				if ( (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') ||
(c >= '0' && c <= '9') )
					cleanText += c;
			}
			for (int x = cleanText.length() - 1; x >= 0; x++)
				invertedText += cleanText.charAt(x);

			System.out.println('"' + text + "\" is " +
(cleanText.equalsIgnoreCase(invertedText)? "" : "not ") + "a palindrome");
			text = read.nextLine();
		}
	}
}
```

6 Vowels ratio

5 points

Introduction

We want to explore if the frequency of consonants in a text is proportional to the frequency of vowels and if this frequency is something that can be characterized. To do so we will evaluate different texts and will compute the ratio of vowels versus consonants. Does this follow a pattern that can be generalized?

Input

The input of the program will be a text to calculate the ratio, ending with the '#'sign. The text will always be shorter than 2000 characters.

Example 1 *(The Iliad Book X, Homer, 800 BC)*

```
Now the other princes of the Achaeans slept soundly the whole night
through, but Agamemnon son of Atreus was troubled, so that he could get
no rest. As when fair Juno's lord flashes his lightning in token of
great rain or hail or snow when the snow-flakes whiten the ground, or
again as a sign that he will open the wide jaws of hungry war, even so
did Agamemnon heave many a heavy sigh, for his soul trembled within him.
When he looked upon the plain of Troy he marvelled at the many
watchfires burning in front of Ilius, and at the sound of pipes and
flutes and of the hum of men, but when presently he turned towards the
ships and hosts of the Achaeans, he tore his hair by handfuls before
Jove on high, and groaned aloud for the very disquietness of his soul.
In the end he deemed it best to go at once to Nestor son of Neleus, and
see if between them they could find any way of the Achaeans from
destruction. He therefore rose, put on his shirt, bound his sandals
about his comely feet, flung the skin of a huge tawny lion over his
shoulders- a skin that reached his feet- and took his spear in his
hand.#
```

Example 2 *(The lord of the rings, J.R.R Tolkien, 1955)*

```
I don't like anything here at all. said Frodo, step or stone, breath or
bone. Earth, air and water all seem accursed. But so our path is laid.
Yes, that's so, said Sam, And we shouldn't be here at all, if we'd known
more about it before we started. But I suppose it's often that way. The
brave things in the old tales and songs, Mr. Frodo, adventures, as I
used to call them. I used to think that they were things the wonderful
folk of the stories went out and looked for, because they wanted them,
because they were exciting and life was a bit dull, a kind of a sport,
as you might say. But that's not the way of it with the tales that
really mattered, or the ones that stay in the mind. Folk seem to have
been just landed in them, usually their paths were laid that way, as you
put it. But I expect they had lots of chances, like us, of turning back,
only they didn't. And if they had, we shouldn't know, because they'd
have been forgotten. We hear about those as just went on, and not all to
a good end, mind you; at least not to what folk inside a story and not
```

```
outside it call a good end. You know, coming home, and finding things
all right, though not quite the same; like old Mr Bilbo. But those
aren't always the best tales to hear, though they may be the best tales
to get landed in! I wonder what sort of a tale we've fallen into? I
wonder, said Frodo, But I don't know. And that's the way of a real tale.
Take any one that you're fond of. You may know, or guess, what kind of a
tale it is, happy-ending or sad-ending, but the people in it don't know.
And you don't want them to. #
```

Output

The program must output: the number of consonants, the number of vowels and the ratio of between them (vowels / consonants). If the ratio cannot be computed, the word "*Infinite*" should be written instead.

Example 1

```
Consonants: 546
Vowels: 327
Ratio: 0.598
```

Example 2

```
Consonants: 742
Vowels: 454
Ratio: 0.612
```

Solution

```
import java.util.Scanner;

public class Prob06 {

    public static void main(String[] args) {
     String word;
     int vowels=0;
     int consonants=0;
     double ratio;
     char caracter;
     boolean end=false;
     Scanner read=new Scanner(System.in);
     word = read.next();

     //Operaciones
     while (!end){ //until we find the #
        for (int x=0;x<word.length();x++){ //analyzing
            caracter=word.charAt(x);
            if ((caracter == 'a' ) || (caracter == 'A' ) || (caracter == 'e' )
|| (caracter == 'E' ) || (caracter == 'i' ) || (caracter == 'I' ) || (caracter
== 'o' )  || (caracter == 'O' ) || (caracter == 'u' ) || (caracter == 'U' ))
                vowels++;
            else if (((caracter >= 'a' ) && (caracter <= 'z' )) || ((caracter
>= 'A' ) && (caracter <= 'Z' )))
                consonants++;
            if (caracter == '#' ) end=true;
        }
        if (!end) word = read.next(); //we continue
     }
     ratio = (double)vowels / consonants;

    //results
    System.out.println("Consonants: " + consonants);
    System.out.println("Vowels: " + vowels);
        if (consonants==0)
            System.out.println("Ratio: Infinite");
        else
            System.out.println("Ratio: " + ratio );
    }

}
```

7 DNA reverse complement

5 points

Introduction

DNA (deoxyribonucleic acid) is the molecule that encapsulates all the information of the life species. The encoding is done through four nucleotides: adenine, thymine, cytosine and guanine that are usually abbreviated as A, T, C, G. In each position of one DNA strand, we can have just one of these nucleotides and the paired strand codifies the opposite base (A links with T, C with G) so the molecule encodes two bits per base pair.
The complement of a DNA sequence is the sequence you get by replacing each base by its pair:
For each base, their corresponding pair is:

G → C
C → G
T → A
A → T

An equivalent protein, called reverse complement, is obtained if you find the complement of a sequence and then reverse it.
Your task is to write a program that finds the reverse complement of a DNA sequence.

Input

The input of the program will be a list of sequence, ending with a '. '. Sequences will always be shorter than 50 nucleotides (expressed as characters). None of them will be longer than 500 elements.

```
AAAACCCGGT
GCGCTTA
.
```

Output

The program must output the reverse complement of the sequences.

```
ACCGGGTTTT
TAAGCGC
```

Solution

```
public class Prob07 {

    public static void main(String[] args) {

    String text;
    String textconverted;
    String textinverted;
    int x,j;
    int width;
    Scanner read=new Scanner(System.in);
    text=read.nextLine();
    while (!text.contentEquals(".")){
        textinverted="";
        textconverted="";
        for (x=0; x<text.length(); x++){
            switch (text.charAt(x))
            {
                case 'A':
                   textconverted = textconverted + 'T'; break;
                case 'T':
                   textconverted = textconverted + 'A'; break;
                case 'C':
                   textconverted = textconverted + 'G'; break;
                case 'G':
                   textconverted = textconverted + 'C'; break;
                default: break;
            }
        }
        j=x;
        for (x=j-1; x>=0; x--)
            textinverted = textinverted + textconverted.charAt(x);
        System.out.println(textinverted);
        text=read.nextLine();
    }
    }

}
```

8 The image histogram

6 points

Introduction

An image histogram is the distribution of pixels for each tonal value in an image.
Image histograms are present on many modern digital cameras. Photographers can use them as an aid to show the distribution of tones captured, and whether image detail has been lost to blown-out highlights or blacked-out shadows.
Your task is to write an algorithm to calculate the histogram of a given grayscale image.
The input to your program will be a ***W by H*** image represented by the grayscale values of every pixel in it. The pixel values will be in the range 0-255, which corresponds to the black-white range. You can assume that the number of pixels in an image can be represented using a 32 bit integer.
The output of your program will be a list of 256 integers. Each one will count the number of pixels that appear in the input with the same value as the position of that integer in the output list. For example, the first integer in the output list will represent the number of pixels with the value 0, the second integer will represent the number of pixels with the value 1, and so on…

Input

The input of the program will be as follows, the first input line contains two unsigned integer values that represent the image width (***W***) and height (***H***) in pixels. Following that, we will have ***H*** input lines each of them containing ***W*** integer values that represent the gray value of the pixel.

```
4 3                 ← this is a 4 by 3 image.
1 2 1 0             ← image row 1
0 0 0 5             ← image row 2
255 5 255 255       ← image row 3
```

Output

The program must output the histogram in a 16x16 matrix that is read from left to right and from tops to bottom.

```
4 2 1 0 0 2 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 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 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 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 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 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 0 0 0 0 0 0 0 0 0 3
```

Solution

```
import java.util.Scanner;

class Prob08{
   public static void main(String[] args) {

         Scanner read=new Scanner(System.in);

         int i,j;
         int [] pixels;
         int [] appearing_pixel;
         pixels = new int[256];
         appearing_pixel= new int[256];
         int width, height;


         width=read.nextInt();
         height=read.nextInt();
         for (i=0; i<256; i++)
             pixels[i]=0;
       // number of pixels
          for (i=0; i<256; i++)
             appearing_pixel[i]=0;
         i=0;
         for (j=0; j<width*height; j++) {
                 pixels[i]=read.nextInt();
                 appearing_pixel[pixels[i]]++;
                 i++;
         }

     i=0;
     while (i<256) {// 256
         for (j=0; j<16; j++){
             System.out.print(appearing_pixel[i]);
             i++;
         }
         System.out.println();
     }
}
}
```

9 Acrostic

6 points

Introduction

An acrostic is a poem or other form of writing in which the first letter, syllable or word of each line, paragraph or other recurring feature in the text spells out a word or a message.

Input

The input of the program will be the full path to read the file that contains the txt file that has to be read. The txt files for these examples are available to be downloaded from the codewars server.

Be careful to use the correct path to your files in your operating system!

The path must be in the format of the computer you are running, for example:
For Windows: `c:\Users\codewars\Documents\celestina.txt`
For Linux: `/users/codewars/Documents/celestina.txt`
For MacOS: `/users/codewars/Documents/celestina.txt`

Files included

`acrostic.txt`
Contains the poem “An acrostic” by Edgar Allan Poe, written in 1829.

`celestina.txt`
Contains the prologue of the Spanish book “La Celestina”, written in 1499 by Fernando de Rojas. This is an interesting acrostic, since the author remained unknown until it was discovered that the piece of the book in this file contains the name of the author.

Output

The program must output the first character of each line in the text.

Output for acrostic.txt

```
ELIZABETH
```

Output for celestina.txt

```
ELBACHJLERFERNANDODEROYASACABOLACOMEDIADECALYSTOYMELYVEAYFVENASCJDOENLAP
VEVLADEMONTALVAN
```

Translated from old Spanish: The high school graduate Fernando de Rojas finished the comedy about Calisto and Melibea and was born in La Puebla de Montalbán (El bachiller Fernando de Rojas acabó la comedia de Calisto y Melibea y fue nacido en la Puebla de Montalbán)

Solution

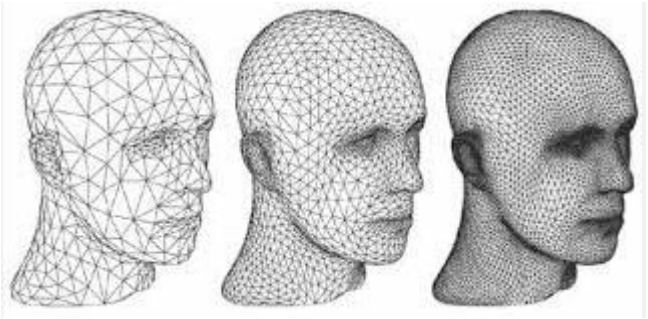
```
import java.io.*;
import java.util.Scanner;

class Prob09{
   public static void main(String [] arg) {
      File fil;
      FileReader fr = null;
      BufferedReader br;
      String file_name;
      Scanner leer=new Scanner(System.in);
      try {
         // Opening the file and creacting the BufferedReader
         file_name=leer.next(); //reading a text
         fil = new File (file_name);
         char caracter;
         fr = new FileReader (fil);
         br = new BufferedReader(fr);

         // reading a file
         String linea;
         while((linea=br.readLine())!=null){
            caracter=linea.charAt(0);
            System.out.print(caracter);

         }
         System.out.println();
      }
      catch(Exception e){
         e.printStackTrace();
      }finally{
         // we close the file
         try{
            if( null != fr ){
               fr.close();
            }
         }catch (Exception e2){
            e2.printStackTrace();
         }
      }
   }
   }
```

10 Tessellating

7 points

Introduction

In order to efficiently show a 3D model to a user it is highly recommendable to use a graphics card that supports either OpenGL or Direct3D standards.
These rendering engines need to be fed, among other things, with a list of triangles in the 3D space of the model that will be shown to the user. The process of transforming a 3D model in a list of triangles is called tessellation.



The algorithm proposed to be implemented will be used to tessellate a 2D convex* polygon.
Write a program that receives a list of 2D vertices, which configure a convex polygon (the last vertex is the same as the first one but it will be provided as part of the input), and returns the list of triangles to be sent to OpenGl or Direct3D.

Example:

Input

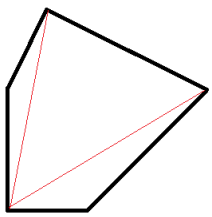
The input of the program is the list of 2D vertices.

```
0,0
2,0
5,3
1,5
0,3
0,0
```

Output

The output of the program is the list of triangles. Each line displays the list of vertices that create the triangle.

```
0,0; 2,0; 5,3
0,0; 5,3; 1,5
0,0; 1,5; 0,3
```



** In a convex polygon, all interior angles are less than or equal to 180 degrees.*

Solution

```
import java.util.Scanner;

class vertex
{
    int x;
    int y;
}

public class Prob10 {

   public static void main(String[] args){
       int i,j;
       boolean fin=false;
       String text;
       int num_vertexs;
       vertex[] poligon;
       poligon=new vertex[100]; // pointing null
       Scanner read=new Scanner(System.in);

       i=0;
       poligon[i]=new vertex();
       text=read.nextLine(); // reading vertex
       poligon[i].x=Character.getNumericValue(text.charAt(0));
       poligon[i].y=Character.getNumericValue(text.charAt(2));
       while (!fin){ // creating triangles
            i++;
            poligon[i]=new vertex();
            text=read.nextLine();
            poligon[i].x=Character.getNumericValue(text.charAt(0));
            poligon[i].y=Character.getNumericValue(text.charAt(2));
            fin=((poligon[i].x==poligon[0].x) && (poligon[i].y==poligon[0].y));
       }
       num_vertexs=i;
       for (i=0; i<num_vertexs-2; i++){
           j=i+1;
           System.out.print(poligon[0].x+","+poligon[0].y+"; ");
           System.out.print(poligon[j].x+","+poligon[j].y+"; ");
           System.out.println(poligon[j+1].x+","+poligon[j+1].y);
        }
   }
}
```

11 My family's vineyard

7 points

Introduction

Your family have a big vineyard and are producing their annual wine. They have a certain amount of sizes of jars and would like to distribute using the biggest jars possible. They need your help to write a program to distribute the wine into the jars.

Input

The first line of the input of the program will be the set of available sizes of the jars in ascending order. There will always be a maximum of 5 sizes and each size will always be smaller than 25. To make sure wine can always be distributed, even if it does not exist, you must assume that size 1 exists.
The rest of the lines will be the amounts of wine to be distributed, always smaller than 256. The input will end with a zero.

```
1 7 15
36
53
10
0
```

Output

The program must distribute the amount of litres produced in the provided jars, using the biggest jars you can, even if it means not using the minimum amount of total jars.

```
36 litres require 2 jar(s) of 15, 6 jar(s) of 1
53 litres require 3 jar(s) of 15, 1 jar(s) of 7, 1 jar(s) of 1
10 litres require 1 jar(s) of 7, 3 jar(s) of 1
```

Solution

```
// my_familiys_vineyard.cpp : Defines the entry point for the console
application.
//

#include <tchar.h>
#include <iostream>
#include <string>
#include <sstream>

const int MaxJars = 5;

void read_jars_from_input(int jar_sizes[], int &total_jars)
{
    std::string input_jars;
    std::getline(std::cin, input_jars);
    // Trim the string in case there are spaces at the beginning or the end
(this is not strictly necessary as the input should be correct)
    size_t first = input_jars.find_first_not_of(' ');
    size_t last = input_jars.find_last_not_of(' ');
    input_jars = input_jars.substr(first, (last - first + 1));

    // Prepare the jars array
    std::stringstream stream_jars(input_jars);
    while (total_jars < MaxJars && !stream_jars.eof())
        stream_jars >> jar_sizes[total_jars++];
}

int _tmain(int argc, _TCHAR* argv[])
{
    int jar_sizes[MaxJars];
    int total_jars = 0;
    // Read the sizes of the jars from the input
    read_jars_from_input(jar_sizes, total_jars);

    // Read amount of liters to distribute
    int wine;
    std::cin >> wine;
    // While we get wine to distribute from the input let's work on it
    while (wine != 0)
    {
        std::cout << std::to_string(wine) << " liters require ";
        int jar = total_jars - 1;
        // While we have not distributed the wine we keep trying different
jars, starting always
        // from the largest one (input is sorted otherwise the algorithm would
not work).

        // We are sure this iteration will finish because we are guaranteed
```

```
        // that the last jar has ALWAYS 1 liter capacity, otherwise the
algorithm may infinitelly loop
        while (wine != 0 && jar >= 0)
        {
            int total_jars = wine / jar_sizes[jar];
            wine %= jar_sizes[jar];

            if (total_jars > 0)
                std::cout << std::to_string(total_jars) << " jar(s) of " <<
std::to_string(jar_sizes[jar]);

            if (total_jars > 0 && wine != 0)
                std::cout << ", ";

            --jar;
        }

        std::cout << std::endl;
        // Read next amount of wine to be distributed
        std::cin >> wine;
    }

     return 0;
}
```

12 The simplest ATM

8 points

Introduction

You have been asked to write a program for a simple ATM (for "automatic teller machine", a machine that lets you withdraw money from your bank account using a credit/debit card). This ATM has the following rules:

- Only 5-euro notes are available. This means that transactions are only accepted if the desired quantity is a multiple of 5.
- Each transaction has a fee of 0.50 euros that must be deducted.
- If a transaction is not possible, the account balance will remain the same.

Input

The first line of the program will be the initial balance of the account (less than 1000 euros) and a series of transactions, ending with a zero.

```
120.0
40.0
32.0
20.0
80.0
0
```

Output

The program must output the enquiry and the final status of the balance after each transaction, one line each.

```
40.0 79.50
32.0 79.50
20.0 59.00
80.0 59.00
```

Detailed explanation

The first result is 79.50 which is 120.00 – 40.00 – 0.50 (fee) = 79.50
The second result is still 79.50 because 32 is not a multiple of 5 and, therefore, the transaction is not valid.
The third result is the result of a 20 euros valid transaction: 79.50 – 20.00 – 0.50 = 59.00
The fourth result is still 59.00 because there are no funds for a 80.00 euro transaction.

Solution

```
import java.util.Scanner;

public class prob12 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int balance = (int) (sc.nextDouble() * 100);
        sc.nextLine();
        int withdrawal;

        String lin;
        while (true) {
            lin = sc.nextLine();
            withdrawal = (int) Math.round(Double.parseDouble(lin) * 100.0);

            if (withdrawal == 0)
                break;

            if ((withdrawal % 500) == 0) // Valid operation so far.
            {
                if (balance >= (withdrawal + 50)) // Valid operation so far.
                    balance -= (withdrawal + 50);
            }
            System.out.println(lin + " " + String.format("%.2f", (balance /
100.0)));
        }

    }

}
```

13 A night at the karaoke

8 points

Introduction

Friends that like singing meet together at someone's home to do a karaoke session. Anyone asks at the beginning for the songs they want to sing. As they sing for hours and hours, they do not want to repeat any song.
Your friend Sing-Chang has asked you to write a program for this Saturday's big party at his home. This program must detect repeated songs and how many times they appear.

Input

The first line of the input contains a number N with the total number of songs (N < 50).
Each line contains the name of a song in lowercase letters a-z. Each song has a maximum length of 80 letters including blanks.

Example 1

```
5
born in the usa
destiny
roxanne
message in a bottle
born in the usa
```

Example 2

```
6
yesterday
hey jude
like a rolling stone
yesterday
hey jude
yesterday
```

Output

The output of the program must be the list of songs, in order of request, without any repetition and the amount of times each of them is requested.

Example 1

```
2 born in the usa
1 destiny
1 roxanne
1 message in a bottle
```

Example 2

```
3 yesterday
2 hey jude
1 like a rolling stone
```

Solution

```
#include <string>     // std::string
#include <iostream>   // std::cin, std::stdl
#include <cstdint>    // uint32_t
#include <functional> // std::hash
#include <vector>     // std::vector

// A song is composed of:
// key:     to represent its uniqueness in the vector
// counter: to represent the number of time it appears
// title:   to represent its unique name
struct song_t {
    uint32_t key;
    uint32_t counter;
    std::string title;
};

int main(int argc, char** argv) {
    // Use a vector to store the songs that have been played in the karaoke
    // A vector is used (instead of a map) because we need to print the songs
    // in the exact order that they have appeared
    std::vector<song_t> list_of_songs;

    // Use a hash function to obtain a unique numeric value based on the song
name
    // This enables us to avoid comparing strings, which is slow compared to
numbers
    std::hash<std::string> hash_function;

    // Read the number of songs to parse from standard input
    std::string line;
    std::getline(std::cin, line);
    uint32_t songs = std::stoi(line);

     // Parse the given number of input songs, one by one
     for (uint32_t i = 0; i < songs; i++) {
          // Read the song name fron standard input
          std::string title;
          std::getline(std::cin, title);

          // Calculate the hash value of the song title
          uint32_t key = hash_function(title);

          // Try to find the key in the song's vector
          auto it = list_of_songs.begin();
          while (it != list_of_songs.end()) {
               // If song is found, increment counter of times that has been
played
               if (it->key == key) {
                    it->counter++;
                    break;
               } else {
                    // Otherwise, go to next song
                    ++it;
```

```
                }
            }

            // If song is not found in the vector, create a new song and append
it at the end
            if (it == list_of_songs.end()) {
                song_t song{key, 1, title};
                list_of_songs.push_back(song);
            }
        }

        // Print all the songs with their corresponding counter
        for (auto it = list_of_songs.begin(); it != list_of_songs.end(); ++it) {
            std::cout << it->counter << " " << it->title << std::endl;
        }

    return 0;

}
```

14 Ave, Caesar!

9 points

Introduction

Julius Caesar is visiting us from the past and he is giving us instructions to build a time machine like his. Our engineers are finding that his instructions include numbers written in Roman. Your task is to write a program that helps them understand these numbers.

As you should already know, Romans used a different numbering system, based on letters and their position. Certain numbers had an assigned letter and the basic rules are:

- If a smaller letter comes afterwards, it adds
- If a smaller letter comes beforehand, it subtracts

Basic characters are:

I is 1
V is 5
X is 10
L is 50
C is 100
D is 500
M is 1000

For instance, in XI the I adds its value to the X because it comes after, so it means 11. In IX the I subtracts its value from the X because it comes before, so it means 9.

Input

The input of the program is a set of Roman numbers ending with a zero. Mmmm, wait! Romans did not have a zero!!! So let's use a dot (.) for the end. The maximum number is 3999 and the longest sequence is the one that produces number 3888 (MMMDCCCLXXXVIII).

```
VII
XLIX
CXX
MCMXCII
MMMDCCCLXXXVIII
MMXV
.
```

Output

The program must output the corresponding Arabic numbers.

```
7
49
120
1992
3888
2015
```

Solution

```
// ave_caesar.cpp : Defines the entry point for the console application.
//

#include <tchar.h>
#include <iostream>
#include <string>
#include <climits>

int get_value(char c)
{
    switch (c)
    {
    case 'I': case 'i':
        return 1;
    case 'V': case 'v':
        return 5;
    case 'X': case 'x':
        return 10;
    case 'L': case 'l':
        return 50;
    case 'C': case 'c':
        return 100;
    case 'D': case 'd':
        return 500;
    case 'M': case 'm':
        return 1000;
    default:
        return 0;
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    std::string roman_number;

    std::cin >> roman_number;
    while (roman_number != ".")
    {
        int number = 0;
        int previous_val = INT_MAX;
        for (std::string::iterator it = roman_number.begin(); it !=
roman_number.end(); ++it)
        {
            // We assume that the input is a correct roman numbers so we don't
check its correctness
            int current_val = get_value(*it);

            // We just keep adding values,
            if (current_val <= previous_val)
            {
```

```
                number += current_val;
                previous_val = current_val;
            }
            // unless the previous character was smaller, which means we did
not deal with the previous character correctly
            else
            {
                // Let's fix this
                number -= previous_val;

                // Add now add the correct quantity
                number += (current_val - previous_val);

                previous_val = INT_MAX;
            }
        }

        // Print the result
        std::cout << number << std::endl;

        // Get next number
        std::cin >> roman_number;
    }

      return 0;
}
```

15 The green internet café

9 points

Introduction

You want to open an internet café where users can book in advance how many hours they want to use your computers. Your shop will be environmental friendly and you want to make sure you only switch on the computers that are required for the day.
A reservation consists of an arrival and a departure time (note that users can only book entire hours, no minutes are provided). Note that if a customer arrives at the same time another one leaves, they are not considered to be in the shop at the same time, for example, if a customer stays from 1 to 2 and another one from 2 to 3, only 1 computer is needed, since they do not stay at the same time.

Input

The first line of the input will be the amount of different customers for the day. The rest of the lines will be the arrival and departure hour for each customer.

Example 1

```
3
1 4
2 5
3 6
```

This means there will be 3 customers, the first one arriving at 1 and leaving at 4, the second one arriving at 2 and leaving at 5 and the third one arriving at 3 and leaving at 6.

Example 2

```
5
1 2
2 3
3 4
4 5
5 6
```

Example 3

```
7
13 19
6 18
5 6
8 9
2 9
10 11
12 15
```

Output

The number of required computers for each day.

Example 1

```
3
```

Example 2

```
1
```

Example 3
3

Solution

```
#include <fstream>
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <algorithm>

using namespace std;

static const int hoursOfTheDay = 24;

int main(int argc, char ** argv)
{
    //Initialize a vector representing the hours of the day
    //where we will increase every reserved hour
    std::vector<int> day(hoursOfTheDay, 0);

    //Variable for reading every input line
    string line;

    //Reads the total computer reservations
    //We use the stringstream to separate the possible values on the line
    int totalEntries= 0;
    getline(cin, line);
    stringstream totalReservations(line);
    totalReservations >> totalEntries;

    //Variables for reading the arrival and departure times
    int arrivalTime, departureTime;

    //Loop that reads every arrival and departure values
    for (int i = 0; i < totalEntries; i++){

        getline(cin, line);
        stringstream hours(line);

        hours >> arrivalTime;
        hours >> departureTime;

        //add an hour to our day vector in the range of arrival and departure
times
        for (int hour = arrivalTime; hour <= departureTime; hour++)
            day.at(hour)++;
    }

    //Checks the hour with more reservations, that will be the maximum number
of
    //computers that we need
    //result variable in this case, has the position of the vector with the
```

```
    //maximum value
    vector<int>::iterator result;
    result = std::max_element(day.begin(), day.end());

    //Shows the result on the screen by locating the position with the maximum
number
    cout << day.at(distance(day.begin(), result)) << "\n";

    return 0;
}
```

16 Permutes

10 points

Introduction

A permute of a number is another number whose digits are the same in a different order.
You have to write a program that calculates all the different permutes for a specific number.

Input

The input of the program is a list of numbers, ending with a zero.

```
135
2468
2442
0
```

Output

The output of the program is the list of permutes of each number, ordered from smallest to biggest.
Numbers cannot be repeated so be careful with those input numbers whose digits are repeated.

```
135
153
315
351
513
531
2468
2486
2648
2684
2846
2864
4268
4286
4628
4682
4826
4862
6248
6284
6428
6482
6824
6842
8246
8264
8426
8462
8624
8642
```

```
2244
2424
2442
4224
4242
4422
```

Solution

```
import java.util.*;

public class prob16 {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);

		while(true)
		{
			String line = sc.nextLine();
			if ("0".compareTo(line)==0)
				break;

			char[] digits = line.toCharArray();
			Arrays.sort(digits);

			do System.out.println(new String(digits));
			while (next_permutation(digits));

		}

	}

	static boolean next_permutation(char[] p) {
		for (int a = p.length - 2; a >= 0; --a)
		  if (p[a] < p[a + 1])
		    for (int b = p.length - 1;; --b)
		      if (p[b] > p[a]) {
		        char t = p[a];
		        p[a] = p[b];
		        p[b] = t;
		        for (++a, b = p.length - 1; a < b; ++a, --b) {
		          t = p[a];
		          p[a] = p[b];
		          p[b] = t;
		        }
		        return true;
		      }
		  return false;
		}

}
```

17 Currency converter

10 points

Introduction

You have to write a simple program that helps convert between currencies, using Euros as the main currency.

Input

The first part of the input will be the number of different currencies the program will be able to use, followed by their conversion rate to EUR (which will be used as the base currency). Each of the rates that come afterwards will be expressed of the amount of the currency 1 Euro is equivalent to.
Afterwards, the list of conversion requests, always expressed as the amount of the original currency followed by the word 'in' and the target currency. A dot will indicate the end of the input.
In the example, 4 currencies are provided and their equivalence is:

- 1 Euro = 1.13 US Dollars
- 1 Euro = 0.74 Pound Sterling (UK)
- 1 Euro = 1.05 Francs (Switzerland)
- 1 Euro = 7.08 Yen (Chinese)

```
4
1.13 USD
0.74 GBP
1.05 CHF
7.08 CNY
5.00 USD in CHF
2.50 GBP in CNY
2.00 USD in EUR
.
```

Output

The program must output the currency conversions

```
5.00 USD = 4.65 CHF
2.50 GBP = 23.92 CNY
2.00 USD = 1.77 EUR
```

Solution

```
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#include <map>

using namespace std;

int main(int argc, char ** argv)
{
    //Variable for reading every input line
    string line;

    //Reads the total currencies for the conversion table
    int totalCurrencies= 0;
    getline(cin, line);
    // Let's show what the stringstream do :)
    // this structure is intend to split the line string into a buffer with
spaced separated strings
    stringstream totalCurrenciesStream(line);
    // So we can read each space separated value and translate it automatically
to the desired type
    // in this case, totalCurrencies is an integer so this statement will
convert this string to integer
    totalCurrenciesStream >> totalCurrencies;

    //Variables for reading the currency name and the currency conversion to
EUR
    float currencyToEur;
    string currencyName;

    //This is the table structure to store the conversion table, we are using a
map instead a vector
    //because a map structure has the search functions implemented and it is
easy to use.
    //So we have a table like this:
    //  ------------
    //  |EUR - 1.00|
    //  |USD - 1.13|
    //  |CHF - 1.05|
    //  |CNY - 7.08|
    //  ------------
    map<string, float> currencyTable;

    //add the first position the euro value
    currencyTable.insert(pair<string,float>("EUR",1.00));

    //Loop that reads the conversion table from the input
    for (int i = 1; i <= totalCurrencies; i++){

        getline(cin, line);
```

```
        //As we saw before, here we have a good example of how the stringsteam
works
        // p.e. line = 1.13 USD
        stringstream currencySteam(line);
        // Now the stringsteam currencyStream has two strings as follows "1.13"
"USD"

        // So we can assign the first string to currencyToEur that is a float
value converting it automatically
        currencySteam >> currencyToEur;
        // And setting the currencyName = USD
        currencySteam >> currencyName;

      // Store the pair "USD",1.13 into the currencyTable generating the pair
with a casting
        currencyTable.insert(pair<string,float>(currencyName,currencyToEur));
    }

    float valueToConvert = 0.00; //Variable where we will store the input
number desired to be converted
    float valueInEur = 0.00; // this variable will temporaly store the
valueToConvert converted to euros
    float finalValue = 0.00; // This variable will store the final conversion
to the foreing currency

    string localCurrency, foreingCurrency;

    getline(cin, line);

    //These statements fix the output number format to X.XX so we have always
two decimal precision
    cout << fixed;
    cout << setprecision(2);

    //Now the final loop where we are reading the desired conversions and it
will stop when a lonely dot is read
    while (line != ".")
    {
        //Get the conversion desired line that has the following format
        // 5.00 USD in CHF
        stringstream conversionStream(line);
        conversionStream >> valueToConvert; // valueToConvert = 5.00
        conversionStream >> localCurrency;  // localCurrency = USD
        //we do it twice for skipping the "in" word in the defined input
        //so at the end we have the second currency into foreingCurrency
        conversionStream >> foreingCurrency; // foreingCurrency = in
        conversionStream >> foreingCurrency; // foreingCurrency = CHF

        //we need the EUR value fo the valueToConvert so we divide the
localCurrency by the value stored in currencyTable

        valueInEur = valueToConvert / currencyTable.at(localCurrency);

        //Now we have the value in EUR so we multiply for the foreing currency
conversion to get the final value.
```

```
        // The sentence currencyTable.at returs the float value of the record
using foreingCurrency as search pattern
        finalValue = valueInEur * currencyTable.at(foreingCurrency);

        //Shows the correct output.
        cout << valueToConvert << " " << localCurrency << " = " << finalValue
<< " " << foreingCurrency << "\n";

        //Gets the next input line for the loop
        getline(cin, line);
    }

    return 0;
}
```

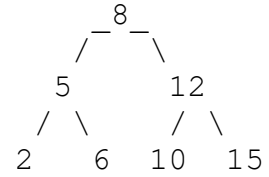
```
import java.util.*;

public class prob17 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        HashMap<String,Double> eurTo = new HashMap<String,Double>();
        int N = sc.nextInt(); sc.nextLine();

        while (N-->0)
        {
            double f = sc.nextDouble();
            String code = sc.next();
            eurTo.put(code, f);
        }
        eurTo.put("EUR", 1.0d);
        sc.nextLine();

        while (true)
        {
            String lin = sc.nextLine();
            if (".".equals(lin))
                break;

            String[] tokens = lin.split(" ");
            double val = Double.parseDouble(tokens[0]);
            double out = val * (eurTo.get(tokens[3]) / eurTo.get(tokens[1]));

            System.out.println(tokens[0]+" "+tokens[1]+" =
"+String.format("%.2f", out)+" "+tokens[3]);
        }
    }
}
```

18 Binary trees

11 points

Introduction

A way to order a sequence of numbers is to do an ordered binary tree. If the tree is made following a given logic it can give us the ordered sequence by processing the tree in in-order traversal.
The logic consists on put the new number in the subtree of the left it is a number smaller or put it into the right subtree if it is greater than the evaluated node. If there is no subtree then a new node is created.
First number is used as the first node (root of the tree).
In-order traversal consists on process first the left child, then the parent and finally the right child.
Given a sequence 8 5 12 2 6 15 10 we have the following tree:



So following this example it is easy to order the sequence.
Other way to read binary trees is post-order in which first are processed the children (left first and then right) and finally the parent.
Write a program that reads a sequence of numbers and builds a binary tree from the numbers as they are read, and then outputs the ordered sequence (in-order) and also the post-order sequence.

Input

A list of numbers ending with -1.
Example 1

```
5
65
37
12
89
32
9
4
-1
```

Output

First the inorder (sorted) numbers and then the post-order sequence.

```
4 5 9 12 32 37 65 89
4 9 32 12 37 89 65 5
```

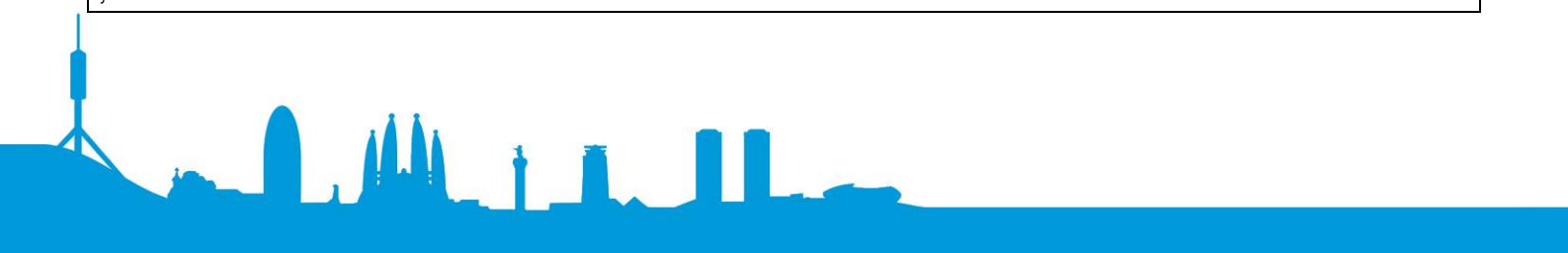
Solution

```
#include "stdlib.h"
#include "stdio.h"

class Node;


// The class Node stores a number and its relationship with the other numbers
(left and right branches)
class Node
{
public:
    Node (int value)
    {
        value_ = value;  // number
        left_ = 0;  // when a node is created it has no branches (it is a leaf)
        right_ = 0; // when a node is created it has no branches (it is a leaf)
    }

    ~Node()
    {
    }

    // get left branch
    Node *getLeft()
    {
        return left_;
    }

    // get right branch
    Node *getRight()
    {
        return right_;
    }

    // set node as left branch
    void setLeft(Node *node)
    {
        left_ = node;
    }

    // set node as right branch
    void setRight(Node *node)
    {
        right_ = node;
    }

    // get the value of current node
    int getValue()
    {
        return value_;
    }
```

```
private:
    Node () {}; // No constructor without value from outside the class
    Node * left_; // left branch (note that a branch or subtree is a tree by
itself)
    Node * right_; // right node. Note that a node contains a tree/subtree
itself
    int value_; // number to store
};


// Insert a new number into an existing tree
void insertNumber (Node *node, int number)
{
    Node *nextNode;

    // If number is less or equal that root number then new value goes into the
left subtree
    if (number <= node->getValue())
    {
        // Take the left subtree
        nextNode = node->getLeft();

      // if there are no numbers then we create a new node an insert as left
subtree
        if (nextNode == 0)
        {
            nextNode = new Node(number); // create new node with the new value
            node->setLeft(nextNode); // insert new node as a left subtree
        }
        else // call again the insertNumber with the left subtree as a tree
(recursive call)
        {
            insertNumber(nextNode, number);
        }
    }
    else // otherwise the number goes into the right subtree
    {
        // Take the right subtree
        nextNode = node->getRight();

      // if there are no numbers then we create a new node an insert as right
subtree
        if (nextNode == 0)
        {
            nextNode = new Node(number);
            node->setRight(nextNode);
        }
        else // call again the insertNumber with the right subtree as a tree
(recursive call)
        {
            insertNumber(nextNode, number);
        }
    }
```

```
}

Node * readTree ()
{
    bool readMore = true;
    int number;


    scanf("%d", &number);

    // First number will be the first node. It is the root of the tree
    Node *root = new Node(number);

    while (readMore)
    {
        scanf("%d", &number);

      //  a new number is inserted into the tree
        if (number >=0)
        {
            insertNumber(root, number); // insert the number into the tree
        }
        else // no more numbers are read
        {
            readMore = false;
        }
    }
    return root;
}

// Inorder prints first left branch, then current node and then right branch
void printInorder(Node* node, bool first)
{
    Node *left;
    Node *right;

    left = node->getLeft();
    right = node->getRight();


    if (left != 0) // only if left is a node it is printed
    {
        printInorder(left, first); // recursive call to print left branch
        first=false;
    }

    // This condition is3yy only because formatting reasons
    // First number to be printed don't need the blank.
    // Other numbers need a blank to separate from previous numbers
    if (first)
    {
        printf("%d", node->getValue());
    }
    else
    {
```

```
        printf(" %d", node->getValue());
    }
    if (right != 0)  // only if right is a node it is printed
    {
        printInorder(right, false);  // recursive call to print right branch
    }

}

// Postorder prints first left branch, then right branch and then current node
void printPostorder(Node* node, bool first)
{
    Node *left;
    Node *right;

    left = node->getLeft();
    right = node->getRight();


    if (left != 0) // only if left is a node it is printed
    {
        printPostorder(left, first);  // recursive call to print left branch
        first = false;
    }
    if (right != 0)  // only if right is a node it is printed
    {
        printPostorder(right, first);  // recursive call to print right branch
        first = false;
    }

    // This condition is3yy only because formatting reasons
    // First number to be printed don't need the blank.
    // Other numbers need a blank to separate from previous numbers
    if (first)
    {
        printf("%d", node->getValue());
    }
    else
    {
        printf(" %d", node->getValue());
    }
}

int main ()
{
    Node *root;

    root = readTree(); // read the tree

    printInorder(root, true); // print inorder
    printf("\n");
    printPostorder(root, true); // print postorder
    printf("\n");

}
```



19 Become a professional bio-scientist

12 points

Introduction

DNA (deoxyribonucleic acid) is the molecule that encapsulates all the information of the life species. The encoding is done through four nucleotides: adenine, thymine, cytosine and guanine that are usually abbreviated by A, T, C, G. In each position of one DNA strand we can have just one of these nucleotides and the paired strand codifies the opposite base (A links with T, C with G) so the molecule encodes two bits per base pair.

Genes are segments of the DNA that code proteins. Finding genes in a DNA is a common and important task in bioinformatics. This is a complex task because there may be skips and mutations in the sequences and DNA molecules are very long. Human DNA contains, for instance, around 2900 million base pairs (Mbp) and is not the longest one, the one of an ameba is 200 times longer!

We are going to simplify a little bit this, the problem you need to solve is to find the first position of a gene coded as nucleotides in a full DNA coded in the same way. The gene may have a maximum of one mutation, so you need to find perfect matches or matches for all nucleotides except one.

Input

First line is the number of genes, followed by one gene each line. Last line is the full DNA to match against. Example:

```
3
TATCGTA
ATCGCA
AAAAAA
CATATCGTAACTGTGC
```

Output

You need to output the first matching position (1 if it matches at first position) or 0 if there is no match. In case of a match, output the number of mutations as well (0 or 1). For the example above, correct output is:

```
3 0
4 1
0
```

Expect longer lengths though, for the genes instead of the 6 or 7 bp of the example, we will use up to 256 bp and for DNA we will use one very short, the DNA of a mitochondria less than 18 kbp.

Solution

```
##############################################################################
##
##
##############################################################################
## (c) Copyright Hewlett-Packard Company, 2015 All rights reserved.
## Copying or other reproduction of this program except for archival
## purposes is prohibited without written consent of Hewlett-Packard Company.
##############################################################################
"""
This program finds the first position of a gene coded as nucleotides (using the
abbreviated chars A, T, C, G) in a full DNA coded in the same way. The gene may
have a maximum of one mutation, so it finds perfect matches or matches for all
nucleotides except one.
"""
__version__ = "0.1"
__date__ = "2015-09-16"
__author__ = "Tanausu Ramirez"

from sys import exit


def read_dna_strand():
    """ Read and check a valid dna strand fron standard input. """
    nucleotides = ['A','C','T','G']
    strand = raw_input()
    for ch in strand.upper():
        if ch not in nucleotides:
            return None
    return strand


def read_input():
    """ Process the input reading of the problem. """
    num_genes = int(raw_input())
    genes = []
    while num_genes:
        strand = read_dna_strand()
        if strand is None:
            exit("ERROR: Wrong gene strand")
        genes.append(strand.upper())
        num_genes -= 1

    dna = read_dna_strand()
    if dna is None:
        exit("ERROR: Wrong DNA strand")

    return genes, dna.upper()


def hamming(str1, str2):
```

```
    """ Compute hamming distance between to equal-length strings """
    if len(str1) != len(str2):
        exit("ERROR: different string len to compute hamming distance")

    hamming_code = [ch1 != ch2 for ch1, ch2 in zip(str1, str2)]
    return sum(hamming_code)


def gen_matching(gen, dna, mutations=1):
    """
    finds the first position of a gene in a dna with up to number of mutations.
    default number is 1.
    Parameters
    ----------
        gen         -- a gen strand
        dna         -- a dna strand
        mutations   -- number of acceptable mutations for the search.
    """
    gen_len = len(gen)
    for i in xrange(len(dna)-gen_len+1):
        d = hamming(gen, dna[i:i+gen_len])
        if d <= mutations:
            return i+1, d
    # Not matching
    return None, None


def do_the_problem():
    genes, dna = read_input()
    for gen in genes:
        idx, mut = gen_matching(gen, dna)
        if idx is None:
            print "0"
        else:
            print "%d %d" % (idx, mut)


if __name__ == '__main__':

    print "Welcome to Become a professional bio-scientist"
    import time
    start = time.clock()
    do_the_problem()
    end = time.clock()
    print "%.2gs" % (end-start)
```

20 The longest kingdom

13 points

Introduction

You have been hired in a newspaper to help in a research they are doing about the Kingdoms in Europe. Your task is to write a program to calculate the days a king or queen have been ruling a country, based on the day they started their kingdom and their death.

WARNING: If you use Python, you cannot use datetime objects.

Input

The first line of the input is the amount of kings and queens, always less than 50. The rest of the input of the program will be a sequence of reign durations, each of them with the name of the King/Queen (first name and order number) and a set of two dates of dates, separated by a space. The dates will be in the format dd/mm/yyyy.

```
4
Edward VII 22/01/1901 06/05/1910
George V 06/05/1910 20/01/1936
Edward VIII 20/01/1936 11/12/1936
George VI 11/12/1936 06/02/1952
```

Output

The program must write the number of days the reign lasted.

```
Edward VII ruled for 3391 days
George V ruled for 9390 days
Edward VIII ruled for 326 days
George VI ruled for 5535 days
```

Solution

```
# 20 The longest kingdom

import sys

#
# Auxiliar function that returns if the year n is leap and false
otherwise
#

def isLeapYear(n):
    if n % 400 == 0:
        return True
    if n % 100 == 0:
        return False
    if n % 4 == 0:
        return True
    else:
        return False
#
# Auxiliar table to storage the days per month [January, February,
March, ... , November, December]
#
daysPerMonth = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

# Variable to store the number of kings
numberOfKings = 0

# Variable to store current king data
currentKingData = []

# Loop reading from standard input
for line in sys.stdin:

    # Consider the case of first line containing the number of kings
    if numberOfKings == 0:
        # Read the first line to get the number of kings
        numberOfKings = int(line)
    else:
        # Rest of lines in the file
        # Divide the line into substrings separated by white space
        # to get the data
        subString = line.split()

        # Put together the name of the king (substring[0]) and
        # its order number (substring[1])
        currentKingData.append(subString[0] + " " + subString[1])

        # Get the start date into three fields [day, month, year]
        currentKingData.append(subString[2])
```

```
        startDate = currentKingData[1].split('/')

        # Split the start date into three integer values: day, month and
year
        startDay   = int(startDate[0])
        startMonth = int(startDate[1])
        startYear  = int(startDate[2])

        # Get the end date into three fields [day, month, year]
        currentKingData.append(subString[3])
        endDate = currentKingData[2].split('/')

        # Split the end date into three integer values: day, month and
year
        endDay   = int(endDate[0])
        endMonth = int(endDate[1])
        endYear  = int(endDate[2])

        # Find out the difference in years, months and days
        years  = endYear - startYear
        months = endMonth - startMonth
        days   = endDay - startDay

        # Let's consider every possible scenario
        if years == 0:
          if months == 0:
            #
            # 1st scenario: Same year and same month, then just count
the days.
            #
            totalDays = days

          else:
            #
            # 2nd scenario: Same year but different months
            #
            sumDays = 0

            # Sum up the days of the months within the range of dates
            for x in range(startMonth, endMonth-1):
              sumDays += daysPerMonth[x]

            # Add the days after first day of starting month.
            sumDays += daysPerMonth[startMonth-1] - startDay

            # Finally add the days of ending month.
            sumDays += endDay

            # Just take into account leap year condition when ending
month is greater
```

```
            # than March to avoid double count it
            if (endMonth >= 3) and isLeapYear(startYear):
              sumDays += 1

            totalDays = sumDays
        else:
          #
          # 3rd scenario: Different years
          #
          sumDays = 0

          # Sum up days between year after starting date and the year
before ending date.
          for x in range (startYear+1, endYear):
            sumDays +=365
            # Add the days corresponding to leap years
            if isLeapYear(x):
              sumDays += 1

          # Count the days for starting year
          for x in range(startMonth, 12):
            sumDays += daysPerMonth[x]

          # Add the days after first day of starting month.
          sumDays += daysPerMonth[startMonth-1] - startDay

          # Just take into account leap year condition when starting
month is greater
          # than March to avoid double count it
          if (startMonth < 3) and isLeapYear(endYear):
            sumDays += 1

          # Count the days for ending year
          for x in range(0, endMonth-1):
            sumDays += daysPerMonth[x]

          # Finally add the days of ending month.
          sumDays += endDay

          # Just take into account leap year condition when ending month
is greater
          # than March to avoid double count it
          if (endMonth >= 3) and isLeapYear(endYear):
            sumDays += 1

          totalDays = sumDays

        # Print the result per each king
        print currentKingData[0] + " ruled for " + str(totalDays) + "
days"
```

```
        # Clean the current king data variable before processing next
line
        currentKingData = []
```

21 Queue theory

14 points

Introduction

There is a complete theory about how queues work. In our case, we would like to create a limited model to study the order in which a bunch of customers will be attended by their cashiers on a supermarket.
The conditions for our experiment are:

- Each cashier spends the same amount of time with each customer (this is just an exercise, not real life).
- There will be a defined number of queues but never less than 2 and never more than 5.
- There is a variable number of customers, never less than 1 and never more than 20 and they are identified by a letter (A,B,C,…)
- The customers are distributed randomly among the different queues.
- If two customers are served at the same time, we would consider that they will be ordered following the queue number they are at (first will be customer in queue 1, second customer in queue 2)

The objective is to give the order in which the customers are attended, for example:

- There are 4 queues:
  - Cashier number 1 spends 3 minutes on each customer
  - Cashier number 2 spends 2 minutes on each customer
  - Cashier number 3 spends 4 minutes on each customer
  - Cashier number 4 spends 1 minutes on each customer
- Customers are distributed as follows:
  - Queue 1 (Cashier 1): Customer A, customer E, customer I
  - Queue 2 (Cashier 2): B, F, J, N
  - Queue 3 (Cashier 3): C, G, L
  - Queue 4 (Cashier 4): D, H, M, O, P, Q

With this input the customers will have been served in the following order and timing:
D (after 1 minute)
B (after 2 minutes on queue 2)
H (after 2 minutes on queue 4)
A (after 3 minutes on queue 1)
M (after 3 minutes on queue 4)
F (after 4 minutes on queue 2)
C (after 4 minutes on queue 3)
O (after 4 minutes on queue 4)
P (after 5 minutes)
E (after 6 minutes on queue 1)
J (after 6 minutes on queue 2)
Q (after 6 minutes on queue 4)
N (after 8 minutes on queue 2)
G (after 8 minutes on queue 3)
I (after 9 minutes)

L (after 12 minutes)

Input

The input will be the number of queues on the first line, followed by the information for each queue, first the time spent by the cashier on a customer, the number of customers on a queue and then the order of the customers (separated by a space). Example:

```
4
3 3 A E I
2 4 B F J N
4 3 C G L
1 6 D H M O P Q
```

Output

The list of served customers ordered by the time spent on the queue separated by spaces:

```
D B H A M F C O P E J Q N G I L
```

Solution

```
##########################################
# 21.     Queue Theory
##########################################
##########################################
#Get Data
##########################################
from collections import deque
queues = int(input())
queueTime = []
queueCustomer = []
customersToServe = 0
for queue in range(queues):
    line = input()
     # line = raw_input()
     # using input() instead of raw_input() crashes with the input
example
     # 4
     # 3 3 A E I
     # 2 4 B F J N
     # 4 3  C G L
     # 1 6 D H M O P Q
     # input() makes the user identify input strings with '"'
     # raw_input() does not care about '"'
     # in order to use input the 2nd to 5th line in the output should be
like
     # "3 3 A E I" instead of 3 3 A E I
     queueData = line.split()
    queueTime.append(int(queueData[0]))
    numberOfCustomers = int(queueData[1])
    queueCustomer.append(deque([]))
    customersToServe += numberOfCustomers
    for customer in range(numberOfCustomers):
        queueCustomer[queue].append(queueData[2 + customer])
##########################################
#Process Data
##########################################
minute = 0
result = ""
while customersToServe > 0:
    minute += 1
    for queue in range(queues):
        if (minute % queueTime[queue] == 0) and
(len(queueCustomer[queue]) > 0):
            result = result + queueCustomer[queue].popleft() + " "
            customersToServe -= 1
##########################################
```

```
#Display results
########################################
print(result)
```

22 GCD and LCM

16 points

Introduction

The greatest common divisor (GCD) of two or more integers is the largest positive integer that divides the numbers without a remainder. The least common multiple (LCM) is the least common multiple of the denominators in a set of fractions.
You need to write a program that calculates the GCD and the LCM of a series of numbers using the factorization method.
This method allows calculating both at the same time and consists on:

1. Find the prime factorization of each number: the prime factorization of an integer is the list of prime factors, along with their multiplicities, that produce the integer. This is unique for each number.
2. Find the GCD by grabbing the terms that all the sequences have in common. In no term is found, GCD is 1.
3. Find the LCD by grabbing all the different terms with their maximum multiplicity

Input

The input of the program is a set of numbers ending with a zero

```
12
36
48
84
0
```

Output

The program must output the prime factorization of each of the numbers, the GCD and the LCM. In the example, the common terms on all the factorization are 2x2x3 so GCD is 12. The different terms with their maximum multiplicity, that is, the set that contains all of them, is 4 repetitions of number 2 (the maximum is in the factorization of 48), 2 repetitions of 3 and 1 repetition of 7, so LCM is 1008 (=2x2x2x2x3x3x7).

```
12 = 2 x 2 x 3
36 = 2 x 2 x 3 x 3
48 = 2 x 2 x 2 x 2 x 3
84 = 2 x 2 x 3 x 7
The greatest common divisor is 2 x 2 x 3 = 12
The least common multiple is 2 x 2 x 2 x 2 x 3 x 3 x 7 = 1008
```

Solution

```
def count_factor_in_dictionary(dictionary, factor):
    """
    Increases the count of a factor in a prime factorization of a
number,
    stored in dictionary.
    """
    if factor in dictionary.keys():
        dictionary[factor] += 1
    else:
        dictionary[factor] = 1


def compute_prime_factorization(n):
    """
    Computes the prime factorization of n, and stores it in a
dictionary.
    The prime factorization is computed by dividing by increasing
integer
    numbers, starting by 2.
    """
    i = 2
    factors = {}

    while i * i <= n:
        if (n % i) != 0:
            i += 1
        else:
            n //= i
            count_factor_in_dictionary(factors, i)

    if n > 1:
        count_factor_in_dictionary(factors, n)

    return factors


def recalculate_gcd(gcd, number_factorization):
    """
    Recalculates the GCD ( Greatest Common Divisor ) based on
    a previous result, and the factorization of a number.
    """
    for factor in gcd.keys():
        if factor in number_factorization.keys():
            gcd[factor] = min(gcd[factor], number_factorization[factor])
        else:
            gcd[factor] = 0
```

```
def recalculate_lcm(lcm, number_factorization):
    """
    Recalculates the LCM ( Least Common Multiple ) based on
    a previous result, and the factorization of a number.
    """
    for factor in number_factorization.keys():
        if factor in lcm.keys():
            lcm[factor] = max(lcm[factor], number_factorization[factor])
        else:
            lcm[factor] = number_factorization[factor]


def get_factorization_string(number_factorization):
    """
    Returns a string representing a number factorization,
    given a dictionary that contains the factorization.
    """
    if all(value == 0 for value in number_factorization.values()):
        return "1"

    string = ""
    for factor, repetitions in number_factorization.items():
        if repetitions != 0:
            for i in range(repetitions):
                string += str(factor)
                string += " x "

    return string[:-3]


def reduce_factorization(factorization):
    """
    Returns the integer number that is represented by the factorization
dictionary.
    """
    partial_result = 1
    for factor, repetitions in factorization.items():
        partial_result = partial_result * factor**repetitions
    return partial_result


if __name__ == "__main__":

    input_finished = False
    first_iteration = True
    gcd = {}
    lcm = {}
    number = int(input())

    while not input_finished:
```

```
        number_factorization = compute_prime_factorization(number)

        print(number, "=",
get_factorization_string(number_factorization))

        if first_iteration:
            # Trivial solutions for the first number.
            first_iteration = False
            gcd = number_factorization.copy()
            lcm = number_factorization.copy()
        else:
            # Recalculate result based on previous result and the
current factorization.
            recalculate_gcd(gcd, number_factorization)
            recalculate_lcm(lcm, number_factorization)

        number = int(input())

        # Iterate while the input is different from 0.
        input_finished = (0 == number)

    print("The greatest common divisor is",
get_factorization_string(gcd), "=", reduce_factorization(gcd))
    print("The least common multiple is", get_factorization_string(lcm),
"=", reduce_factorization(lcm))
```

23 Double auctions

18 points

Introduction

A double auction is an economic mechanism to allocate identical goods from different vendors to multiple buyers. In a double auction, potential buyers submit their bids and potential sellers simultaneously submit their ask prices to an auctioneer. Then an auctioneer chooses some price p that clears the market: all the sellers who asked less than p sell and all buyers who bid more than p buy at this price p. As well as their direct interest, double auctions are reminiscent of Walrasian auction and have been used as a tool to study the determination of prices in ordinary markets. In this exercise we want to implement a double auction market where HP sellers will offer their ink supplies and HP customers will bid for them.

To clear the auction, the M^{th} and $(M + 1)^{st}$ prices are computed, where M is the total number of asks. Conceptually, finding the M^{th} and $(M + 1)^{st}$ prices is simply a matter of sorting the bids and asks in descending order (bigger to smaller), merging them in a single lists of prices, and identifying the M^{th} and $(M + 1)^{st}$ elements in it. The prices between the M^{th} and $(M + 1)^{st}$ bids (inclusively) represent the range of prices for which supply balances demand.

The pricing policy that we will be using is the so named k-pricing policy where **PK** is computed as follows:

$$PK = k * \mathbf{P_{(M+1)}} + (1-k) * \mathbf{P_M}$$

Where **k** is the so-named welfare factor and $\mathbf{P_M}$ and $\mathbf{P_{(M+1)}}$ are the M^{th} and $(M + 1)^{st}$ prices correspondingly.

Input

The input will start with the input of the k welfare factor, followed by the sequence of bids (ended by a -1.0) and followed by a sequence of asks (also ended by a -1.0).

```
0.5       ← this is the value of the welfare factor.
22.1
22.5
22.7
23.0
21.9
20.1
19.8
22.6
-1.0      ← this is the end of the bids, starting asks
22.1
23.6
22.2
22.4
-1.0      ← this is the end of asks
```

Output

The program must output the final price, the asks sold sorted in ascending order (small to big) and the bids sold sorted by descending order (big to small).

```
Price: 22.55
Asks sold: 22.1, 22.2, 22.4
Bids sold: 23.0, 22.7, 22.6
```

Detailed explanation

M = 4 (number of asks)
Sorted bids and asks in descending order.
23.6
23.0
22.7
22.6 -> Mth price
22.5 -> Mth +1 price
22.4
22.2
22.1
22.1
20.1
19.8

K = 0.5
P = (0.5 * 22.5) + ((1 - 0.5) * 22.6) = 22.55
Asks below 22.55 match with bids over 22.55
Asks sold: 22.1, 22.2, 22.4
Bids sold: 23.0, 22.7, 22.6

Solution

```
/* -*- C++ -*- */

////////////////////////////////////////////////////////////////////////////
//
//  A possible solution for Double Action problem.
//
//  \author Josep Giralt (josep.giralt@hp.com)
//  \date   Wed Sep 16 11:45:41 CEST 2015
//
//  (C) Copyright 2015 Hewlett-Packard Inc.
//
////////////////////////////////////////////////////////////////////////////

// Standard includes
//
#include <iostream>
#include <vector>
#include <algorithm>
#include <iomanip>
#include <map>

/** defines the type of element (either a bid or a ask) in the price vector.
 */
typedef enum
{
    BID,        /// the element will be a bid.
    ASK         /// the element will be an ask.
} element_type_t;


/** binary predicate that will be used to sort the price vector in the
descending order.
    \param[in] p1 a reference to the first predicate element to compare.
    \param[in] p2 a reference to the seconf predicate element to compare.
    \return \e true if the \e p1 is strictly greater than \e \p2.
*/
bool descending_sort_predicate(const std::pair<double, element_type_t> &p1,
const std::pair<double, element_type_t> &p2)
{
    return p1.first > p2.first;
}
```

```
/** entry point. we asume that the input parameters will be correct so there is
no need to perform any input parameter checking
 */
int main (int c, char * argv[])
{
    int    m;
    double p;
    double k;
    bool   first;

    // the vector of price elements.
    // Each element in the vector will be a STL pair where \e first is the
price and \e second is the element type (bid or ask).
    //
    std::vector< std::pair<double, element_type_t> > prices;

    // read the k welfare factor.
    //
    std::cin >> k;

    // read the bids.
    //
    std::cin >> p;
    while (p != -1.0)
    {
        prices.push_back(std::make_pair(p, BID));
        std::cin >> p;
    }

    // read the asks.
    //
    m = 0;
    std::cin >> p;
    while (p != -1.0)
    {
        m++;
        prices.push_back(std::make_pair(p, ASK));
        std::cin >> p;
    }

    // sort the all prices in descending order, because we use the \ref
descending_sort_predicate
    // the prices will be sorted in descending order (bigger to smaller).
    //
    std::sort (prices.begin(), prices.end(), descending_sort_predicate);

    // compute the price. note that the vectors starts from 0 so the Mth price
    // is in fact the (M-1) element in the vector. We can safely access to it
    // as we were told that there will be always at least 1 ask and 1 bid.
    //
    p = ((1 - k) * (prices[m-1].first)) + (k * prices[m].first);

    std::cout << "Price: " << std::fixed << std::setprecision(2) << p <<
std::endl;
```

```
    // output the asks traversing the vector in reverse order, we will use a
reverse iterator for that matter.
    //
    first = true;
    std::vector< std::pair<double, element_type_t> >::const_reverse_iterator
a_it;
    std::cout << "Asks sold:";
    a_it = prices.rbegin();
    while ((a_it != prices.rend()) && (a_it->first <= p))
    {
        if (a_it->second == ASK)
        {
            std::cout << (first ? " " : ", ") << std::fixed <<
std::setprecision(2) << a_it->first;
            first = false;
        }
        ++a_it;
    }
    std::cout << std::endl;

    // output the bids traversing the vector in forward order.
    //
    first = true;
    std::vector< std::pair<double, element_type_t> >::const_iterator b_it;
    std::cout << "Bids sold:";
    b_it = prices.begin();
    while ((b_it != prices.end()) && (b_it->first >= p))
    {
        if (b_it->second == BID)
        {
            std::cout << (first ? " " : ", ") << std::fixed <<
std::setprecision(2) << b_it->first;
            first = false;
        }
        ++b_it;
    }
    std::cout << std::endl;

    return 0;
}
```

24 Interstellar

21 points

Introduction

Do you like solving mazes? We are sure you don't, they are too easy to solve, and they can be solved by mice! But this is in part because we make them too simple, we make them in two dimensions to be able to draw them on a 2D surface. But universe is 3D, well, or 4D if we include time.
So we need to write a program to solve a 4D or space-time maze, but remember the fourth dimension is time and time dimension has something special in our universe, you cannot go back in time. Of course, you can change the speed while moving in this space-time maze, this means that when in a specific time slice you can move as much as you want through the 3D space maze or even not move at all before you jump to the next time slice. Traveling in diagonal is not allowed, neither in space, nor in time.
The maze may have multiple solutions, but you need to return the solution that is the shortest in the number of required movements in space.

Input

The first line will be the size of the space time of our maze: Nx, Ny, Nz, Nt, all of them are positive integers. All space-time values are integers and start by 0, if Nx = 2, for example, possible x positions are 0, 1 and 2. Second line is the starting point: Sx, Sy, Sz, St. St will be always 0, start of time. Third line is the ending point: Ex, Ey, Ez, Et. Where Et = Nt – 1, end of time.
After this, there are Nx * Ny * Nz rows, each row start by 3 numbers than are the x, y, z coordinates of one point in space and later there are Nt values representing the evolution in time of this point in the maze, possible values are an asterisk (*) for a wall you can´t cross or an underscore (_) for a space.

Output

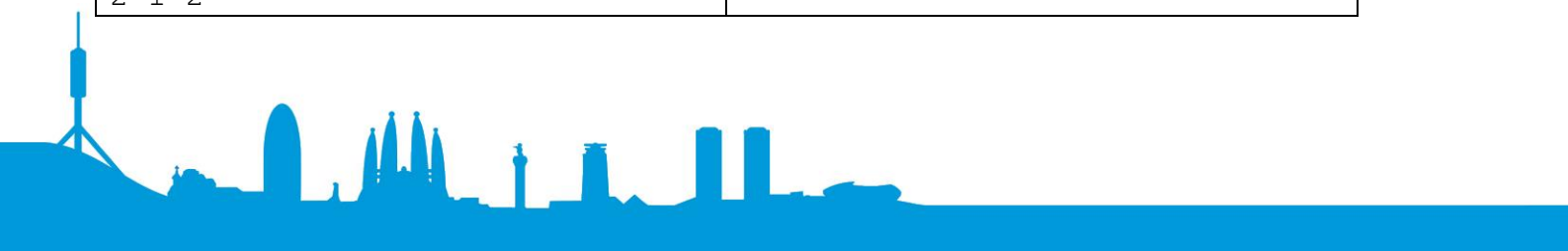
Output the number of steps followed by the path thru space-time in x, y, z, t coordinates.

Input example:

```
3 2 3 4
0 1 1 0
1 1 0 3
0 0 0 **_*
0 0 1 ___*
0 0 2 ___*
0 1 0 *__*
0 1 1 ___*
0 1 2 __**
1 0 0 ****
1 0 1 ****
1 0 2 *___
1 1 0 **__
1 1 1 *_**
1 1 2 ****
2 0 0 ****
2 0 1 ****
2 0 2 ****
2 1 0 **_*
2 1 1 __**
2 1 2 ****
```

Output Example:

```
6
0 1 1 0
0 1 1 1
0 1 1 2
0 1 0 2
1 1 0 2
1 1 0 3
```



Solution

```
import java.util.*;

public class prob24 {
    static int X, Y, Z, T;      // Boundaries
    static int Sx, Sy, Sz, St; // Start position
    static int Ex, Ey, Ez, Et; // End position

    // Original mazes in the timeline.
    static ArrayList<char[][][]> timeline = new ArrayList<char[][][]>();

    // Next moves in the Breath First Search.
    static LinkedList<Pos> moves = new LinkedList<Pos>();

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        X = sc.nextInt();
        Y = sc.nextInt();
        Z = sc.nextInt();
        T = sc.nextInt();

        Sx = sc.nextInt();
        Sy = sc.nextInt();
        Sz = sc.nextInt();
        St = sc.nextInt();

        Ex = sc.nextInt();
        Ey = sc.nextInt();
        Ez = sc.nextInt();
        Et = sc.nextInt();

        for (int t=0;t<T;t++)
            timeline.add(new char[X][Y][Z]);

        // Read the maze and store it in the timeline.
        for (int i=0;i<X*Y*Z;i++)
        {
            int x,y,z;
            x = sc.nextInt();
            y = sc.nextInt();
            z = sc.nextInt();
            String seq = sc.next();
            for (int t=0;t<T;t++)
                timeline.get(t)[x][y][z] = seq.charAt(t);
        }
        sc.close();

        // Add the start position:
        Pos s = new Pos();
        s.x = Sx;
        s.y = Sy;
        s.z = Sz;
        s.t = St;
        s.parent = null;
```

```
        s.cube = timeline.get(St);
        moves.add(s);

        // Breadth First Search (as soon as we find the Exit point for the
first time, we know it's the shortest path in number of steps)
        while (moves.size()>0)
            move(moves.removeFirst());
    }

    static void move(Pos c)
    {
        c.cube[c.x][c.y][c.z] = 'v';  // Mark as visited

        if (c.x==Ex && c.y==Ey && c.z==Ez && c.t==Et) // Found the exit point
:)
        {
            printSolution(c);
            System.exit(0);
        }

        Pos n;
        if (c.t+1<T)  // Attempt to move ahead in time
        {
            n = c.clone();
            n.cube = timeline.get(++n.t);  // Move to the next cube.
            if (validPosition(n)) {n.cube[n.x][n.y][n.z]='a'; moves.add(n);}
        }

        // Attempt to move within the current maze:
        n = c.clone(); n.x++; if (validPosition(n)) {n.cube[n.x][n.y][n.z]='a';
moves.add(n);}
        n = c.clone(); n.x--; if (validPosition(n)) {n.cube[n.x][n.y][n.z]='a';
moves.add(n);}
        n = c.clone(); n.y++; if (validPosition(n)) {n.cube[n.x][n.y][n.z]='a';
moves.add(n);}
        n = c.clone(); n.y--; if (validPosition(n)) {n.cube[n.x][n.y][n.z]='a';
moves.add(n);}
        n = c.clone(); n.z++; if (validPosition(n)) {n.cube[n.x][n.y][n.z]='a';
moves.add(n);}
        n = c.clone(); n.z--; if (validPosition(n)) {n.cube[n.x][n.y][n.z]='a';
moves.add(n);}
    }

    // Maze legend: '_' free, '*' wall, 'a' added to the queue, 'v' visited.
    static boolean validPosition(Pos n)
    {
        if (n.x<0||n.x>=X||n.y<0||n.y>=Y||n.z<0||n.z>=Z) // Check for
boundaries
            return false;
        else if (n.cube[n.x][n.y][n.z]=='_')  // Only allow moving to an empty
space.
            return true;
        else
            return false;
    }
```

```
    static void printSolution(Pos c)
    {
            // Rebuild the path (in inverse order):
            LinkedList<Pos> path = new LinkedList<Pos>();
            Pos p = c;
            while (p!=null)
            {
                path.addFirst(p);
                p=p.parent;
            }

            // Print the length of the pathm and the steps in the right order:
            System.out.println(path.size());
            Iterator<Pos> it = path.iterator();
            while (it.hasNext())
            {
                p = it.next();
                System.out.println(""+p.x+" "+p.y+" "+p.z+" "+p.t);
            }
    }

    // Represents a Position in the multi-dimentional maze(x,y,z,t), its
environment (current cube)
    // and the history behind it (parent position).
    static class Pos
    {
        int x,y,z,t;      // Current position.
        char[][][] cube;  // Current cube configuration (in 't' time).
        Pos parent;       // Previous position.

        public Pos clone()
        {
            Pos n = new Pos();
            n.x = x;
            n.y = y;
            n.z = z;
            n.t = t;
            n.cube = cube;  // Shallow copy to share the same maze status.
            n.parent = this; // 'this' is the predecessor of the cloned node.
            return n;
        }
    }
}
```