Asian MathSci League, Inc (AMSLI)
Website: amsliphil.com
Email address: amsliphil@yahoo.com

Student Copy

AIEP Sendoff HS Session 9

ASSORTMENT OF PROGRAMMING CHALLENGES (v. 1.0)

"Computers are good at following instructions but not at reading your mind."

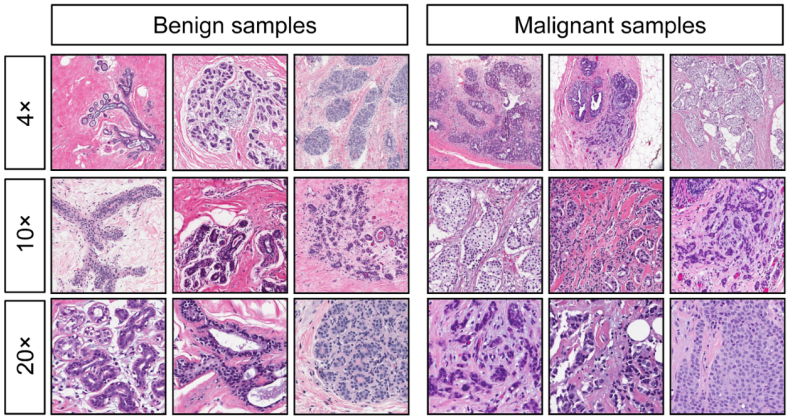
~ Donald Knuth

In this handout, we present some programming challenges. The top priority is always to produce a correct algorithm, written in syntactically and semantically correct Python code. The next step is to overhaul the algorithm with the goal of making it more efficient although this does require a bit more time, insight, and practice.

YOUR SOLUTION MUST STRICTLY ADHERE TO THE SPECIFICATIONS IN EACH PROBLEM. Unless otherwise stated, your program must accept input from the standard input device (keyboard) and display the required output on the standard output device (monitor screen). Failure to follow the instructions will invalidate your submission.

PROBLEM 1: Distance is More than Just a Number

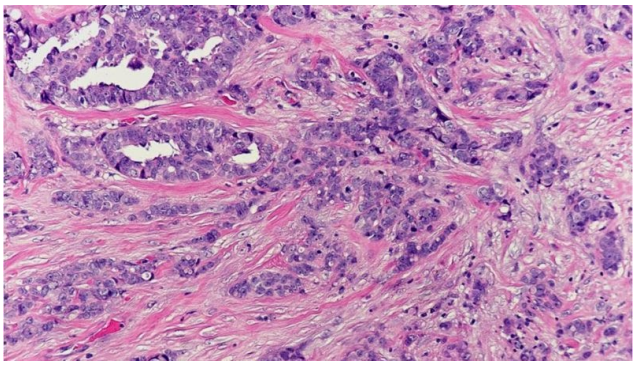
An active area in artificial intelligence research today is *machine learning* — teaching a computer how to "learn on its own" without explicitly laying out rules and conditions. To see how this area is relevant in other disciplines, we take a look at the following scenario:



https://static.packt-cdn.com/products/9781783980284/graphics/3a298fcc-54fb-42c2-a212-52823e709e30.png

Asian MathSci League, Inc (AMSLI)
Website: amsliphil.com
Email address: amsliphil@yahoo.com

This collection of images shows benign (noncancerous) and malignant (cancerous) breast tissues after staining. Suppose that we are given the following picture of a breast tissue and we are tasked to find out whether it is benign or malignant:



https://pathology.jhu.edu/breast/types-of-breast-cancer

We might guess that this a malignant tumor since it looks "similar" to the second picture in the third row of our malignant samples. We are, in fact, correct, and a trained histologist will identify this as *invasive ductal carcinoma*.

Now, one of the goals of machine learning is to allow us to create programs that can automatically detect a possibly cancerous tumor based on photos of breast tissues. For humans, our visual system allows us to recognize "similarity" quite intuitively, but this is not the case for computers that can rely only on cold, hard data.

The solution to this problem is to take a look at some of the features: radius, perimeter, area, concavity, symmetry of the tissue, etc. Then, we represent them as a list of numerical data. We then compare how "similar" these features are against the features of a photo of an actual cancerous tumor (as well as that of a benign sample) to determine whether the photo shows a malignant mass or not.

"Similarity" in machine learning is formalized as the mathematical "distance" between the two images, or two data points in general. Shorter distance means higher similarity. There are various metrics to compute for the distance, and we are going to encounter three of them in this programming challenge.

Prepared by ***Mark Edward M. Gonzales***

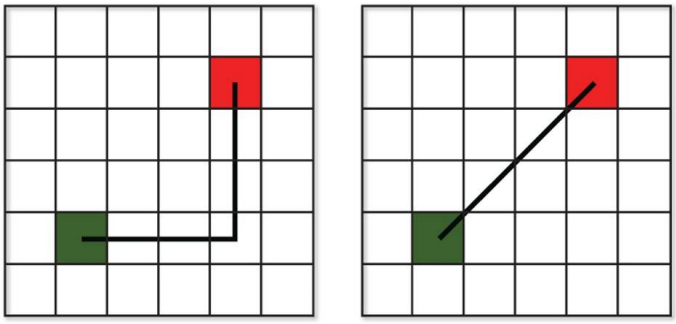
Suppose the features of the first data point are represented using the list $[a_1, a_2, a_3, ..., a_n]$ and those of the second data point are represented using the list $[b_1, b_2, b, ..., b_n]$. Since these features are numerically encoded, we can define the distance metrics as follows:

a. **Euclidean Distance.** Shortest distance between two data points:

$$\sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + (b_3 - a_3)^2 + ... + (b_n - a_n)^2}$$

b. **Manhattan Distance.** Distance between two data points measured along axes at right angles:

$$\left|(b_1 - a_1) + (b_2 - a_2) + (b_3 - a_3) + ... + (b_n - a_n)\right|$$



https://cstopics.github.io/cstopics/artificial-intelligence/search/insearch

Left: Manhattan distance; Right: Euclidean distance

c. **Minkowski Distance of Order p.** Generalization of the two previous distances:

$$\left|(b_1 - a_1)^p + (b_2 - a_2)^p + (b_3 - a_3)^p + ... + (b_n - a_n)^p\right|^{\frac{1}{p}}$$

INPUT

The input consists of four lines:

a. **Line 1:** Number of features of each data point
b. **Line 2:** Integer p denoting the order for calculating the Minkowski distance
c. **Line 3:** Numerically encoded features of the 1st data point (separated by a space)
d. **Line 4:** Numerically encoded features of the 2nd data point (separated by a space)

OUTPUT

The output consists of three lines:

a. **Line 1:** Euclidean distance between the two data points
b. **Line 2:** Manhattan distance between the two data points
c. **Line 3:** Minkowski distance between the two data points

The distances must be displayed with exactly six (6) digits after the decimal point.

SAMPLE RUN

Input	Output
10 4 1.51 2.46 3.2 4.9 1.5 27.912 1.0 0.0 1.0 0.0 2.3 3.84 1.0 14.9 1.5 87.57 0.0 1.0 1.0 0.0	60.567693 76.028000 59.669805

RESTRICTIONS

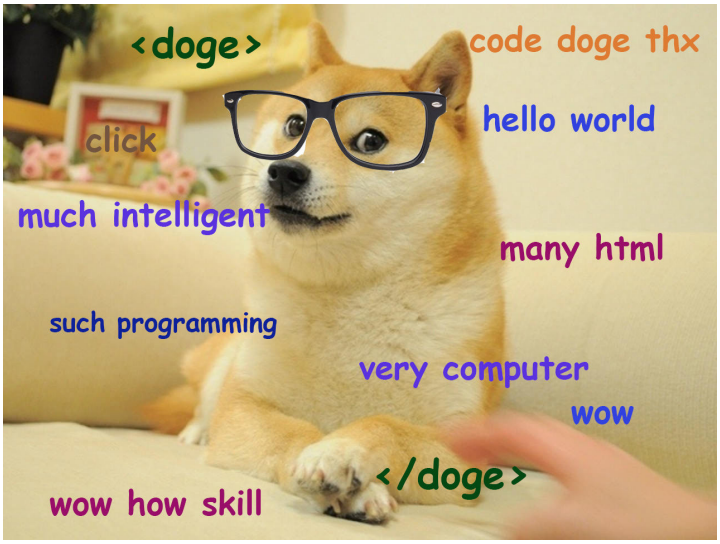
a. Use the built-in `round()` function in displaying the distances with 6 digits after the decimal point.
b. Your solution must take only a single look/pass at the features of the data points. In other words, it must calculate all three distances using exactly one (1) loop only.

PROBLEM 2 STARTS ON THE NEXT PAGE

Asian MathSci League, Inc (AMSLI)
Website: amsliphil.com
Email address: amsliphil@yahoo.com

PROBLEM 2: The H1dden D0ge

"Doge" is a dank meme featuring the image of a Shiba Inu (柴犬, a Japanese dog breed) named Kabosu as it peculiarly sits on a couch like Mona Lisa. First appearing in the personal blog of kindergarten teacher Atsuko Sato on February 2010, and popularized eight months later when a picture was uploaded to a subreddit, this irreverent canine has crossed the meme boundary to lend its name to the cryptocurrency Dogecoin.



https://dogemuchwow.com/web-developer-doge/

One day, Zoe (who is extremely fond of Doge memes), noticed that the words "DOGE" and "KABOSU" contained nine unique letters. This inspired her to form an encryption system featuring these letters. She first took a look at the well-known Morse Code:

Letter	Morse Code	Letter	Morse Code
D	- . .	K	- . -
O	- - -	A	. -
G	- - .	B	- . . .
E	.	S	. . .
		U	. . -

Prepared by ***Mark Edward M. Gonzales***

Since Morse Code is a variable-length encoding scheme (that is, Morse code equivalents have different total numbers of dots and dashes), certain ambiguities can arise. Consider, for instance, the encryption ". - - . . . . . ", which can be interpreted as "ABS" (. - then - . . . then . . .), "EGEES" (. then - - . then . then . then . . .), etc. Morse code transmitters use pauses to resolve this, but Zoe wanted a better solution.

Her friend suggested the use of a fixed-length encoding scheme. However, Zoe feistily disliked this idea since she considered it wasteful and inefficient. She argued that it made better sense if letters that appear more frequently in words (like the vowels "a" and "e") are encoded using fewer symbols so that they can be transmitted more quickly. On the other hand, letters that are relatively rare can accommodate more symbols.

Fueled by her fondness for Doge and for computer science, she diligently worked hard to devise a non-ambiguous variable-length encoding scheme. One day, she came across a paper detailing prefix codes and got inspired to create one for her truncated alphabet:

Letter	Zoe's Code	Letter	Zoe's Code
D	11011	K	11010
O	101	A	0
G	11111	B	11110
E	100	S	1110
		U	1100

In mathematical terms, Cornell University professors Kleinberg and Tardos (2006) defined a prefix code for a set S of letters as "a function γ that maps each letter $x \in S$ to some sequence of zeros and ones, in such a way that for distinct $x, y \in S$, the sequence $\gamma(x)$ is not a prefix of the sequence $\gamma(y)$."

To illustrate Zoe's scheme, consider the word "BAKE", which translates to 11110011010100. The decoding of this "bit string" leaves no room for ambiguity: 11110 (B) then 0 (A) then 11010 (K) then 100 (E). *A baked treat, Doge is pleased.*

INPUT

The input consists of n lines:

a. **Line 1:** Number of bit strings to be decoded (following Zoe's scheme)
b. **Lines 2 to n:** Bit strings to be decoded

OUTPUT

The output consists of $n-1$ lines:

a. **Lines 1 to $n-1$:** Decoded bit strings. If the bit string cannot be decoded using Zoe's scheme, then the word `BAMBOOZLED` is displayed.

SAMPLE RUN

Input	Output
4 111101100111111110 110111100110101001110 1101 0	BUGS DUKES BAMBOOZLED A

PROBLEM 3 STARTS ON THE NEXT PAGE

Asian MathSci League, Inc (AMSLI)
Website: amsliphil.com
Email address: amsliphil@yahoo.com

PROBLEM 3: Fortune Favors the Prime

In Chinese numerology, the number 8 (八, bā) is considered a lucky number due to its phonetic resemblance to the Chinese word for "prosper": 發 (fā). But luck does not equate to fortune — at least in this programming challenge.

In the 20th century, the New Zealand-born social anthropologist Reo Franklin Fortune (once married to the influential anthropologist Margaret Mead, best known for her books on Southeast Asian and South Pacific culture) ventured into the world of mathematics and lent its name to the so-called "Fortunate numbers."

https://almagottlieb.com/category/reo-fortune/

Reo Fortune (rightmost) with fellow anthropologists Gregory Bateson and Margaret Mead

Before going into detail about Fortunate numbers, we first introduce the notion of the primorial. The primorial of a whole number n, denoted as $n\#$, is the product of all the prime numbers less than or equal to n. By convention, $0\# = 1\# = 1$. (Recall that a prime number is a number with exactly two positive factors: 1 and the number itself.)

> Let us compute for $15\#$. The prime numbers less than or equal to 15 are 2, 3, 5, 7, 11, and 13. Taking their product gives the primorial of 15: 30030.

The n^{th} **Fortunate number** is the smallest integer m greater than 1 such that $p_n\# + m$ is a prime number. Note that $p_n\#$ refers to the primorial of the n^{th} prime number.

Let us compute for the 6th Fortunate number. From the previous example, we know that $p_6\# = 2 \times 3 \times 5 \times 7 \times 11 \times 13 = 30030$. The 6th Fortunate number is the smallest integer m greater than 1 such that $30030 + m$ is prime. With some computation, we find that $30030 + 17 = 30047$ is prime. Therefore, the 6th Fortunate number is 17.

In a stroke of mathematical genius and humor, Paul Carpenter introduced the n^{th} **lesser Fortunate number** as the smallest integer m greater than 1 such that $p_{n+1}\# - m$ is a prime number.

Let us compute for the 6th lesser Fortunate number. Be careful since this is related to the primorial of the 7th prime (<u>not</u> the 6th prime): $p_7\# = 2 \times 3 \times 5 \times 7 \times 11 \times 13 \times 17 = 510510$. The 6th lesser Fortunate number is the smallest integer m greater than 1 such that $510510 - m$ is prime. With some computation, we find that $510510 - 29 = 510481$ is prime. Therefore, the 6th lesser Fortunate number is 29.

Test your understanding by trying to compute for the previous and succeeding Fortunate and lesser Fortunate numbers. We list down the first 12 numbers as bases for comparison:

- Fortunate numbers

  3 5 7 13 23 17 19 23 37 61 67 61

- Lesser Fortunate numbers

  3 7 11 13 17 29 23 43 41 73 59 47

You might have noticed that there are duplicates and that there is no particular ordering to the numbers. In this programming challenge, <u>leave them as they are</u>. Therefore, the 10th and 12th Fortunate numbers are both equal to 61.

Prepared by ***Mark Edward M. Gonzales***

We cap off this introduction with the decades-old unresolved conjecture that all Fortunate and lesser Fortunate numbers are prime. The composite 8 may be lucky — but most likely un-Fortunate.

INPUT

The input consists of a single line:

a. **Line 1:** A positive integer n

OUTPUT

The output consists of a single line:

a. **Line 1:** Number of distinct Fortunate numbers (from the 1st until the n^{th} Fortunate number) that also appear as part of the first n lesser Fortunate numbers

SAMPLE RUN

Input	Output
12	5

RESTRICTIONS

a. To test whether a number is prime or not, you are not allowed to use naive trial division (that is, testing possible divisors up to the square root of the number) in the interest of efficiency. Primorials grow at an exceedingly fast rate!
b. You are given the liberty to write your own implementation of any other (faster) primarily test, such as the $6k \pm 1$ optimization discussed in one of our sessions.

PROBLEM 4 STARTS ON THE NEXT PAGE

Asian MathSci League, Inc (AMSLI)
Website: amsliphil.com
Email address: amsliphil@yahoo.com

PROBLEM 4: Narutomaki

One evening, while Hinata was eating ramen with her husband Naruto in Ichiraku Ramen, she fixed her gaze on the *narutomaki* (a cured fish paste that features a distinct pinkish to reddish spiral pattern). For some eccentric reason, the spiral patterns on this Japanese topping took her back to her math classes in the Academy and inspired her to ponder on the properties of number spirals:

37	36	35	34	33	32	31
38	17	16	15	14	13	30
39	18	5	4	3	12	29
40	19	6	1	2	11	28
41	20	7	8	9	10	27
42	21	22	23	24	25	26
43	44	45	46	47	48	49

This figure shows a number spiral with 7 rows and 7 columns. At the middle of the spiral is the first positive integer (1), and the succeeding positive integers are written in such a way that a complete square grid is filled in a counterclockwise spiral fashion

Naruto termed the green cells (together with the central yellow cell) as the main diagonal entries of the spiral. On the other hand, he referred to the blue cells (together with the central yellow cell) as the antidiagonal entries of the spiral. Take a closer look at these entries and try to unearth some interesting properties. Brave *shinobi* may want to prove the veracity of these patterns and explain why they occur.

Anyway, Hinata was more interested in the union sum of the number spiral, defined as the sum of the main diagonal and the antidiagonal entries, with duplicate entries treated as only a single addend. In the number spiral shown, the union sum is equal to the sum of 37, 17, 5, 9, 25, 49, 31, 13, 3, 7, 21, 43, and 1 (which is counted only once): 261. *Dattebayo!*

Prepared by ***Mark Edward M. Gonzales***

INPUT

The input consists of a single line:

a. **Line 1:** Number of rows of the number spiral (which is also equal to the number of columns). For simplicity, assume that this is always an odd number. In addition, the number spiral is generated in the manner described in the given example.

OUTPUT

The output consists of a single line:

a. **Line 1:** Union sum of the number spiral

SAMPLE RUN

Input	Output
7	261

BONUS CHALLENGE

Create a solution that runs in $O(1)$ time. Interested trainees may also want to take a look at Part II, Problem 2 of the 18th Philippine Mathematical Olympiad Area Stage, which inspired this programming challenge.

PROBLEM 5 STARTS ON THE NEXT PAGE

Asian MathSci League, Inc (AMSLI)
Website: amsliphil.com
Email address: amsliphil@yahoo.com

PROBLEM 5: Multiplying by Rotation

(1989 Europe Northwestern ACM-ICPC)

ACM-ICPC is the Association for Computing Machinery – International Collegiate Programming Challenge

WARNING

Not all numbers in this problem are decimal numbers!

Multiplication of natural numbers in general is a cumbersome operation. In some cases however the product can be obtained by moving the last digit to the front.

Example: 179487 * 4 = 717948

Of course this property depends on the number system you use, in the above example we used the decimal representation. In base 9 we have a shorter example:

17 * 4 = 71 (base 9)

as (9 * 1 + 7) * 4 = 7 * 9 + 1

INPUT *(rotamult-input.txt)*

The input for your program is a text file. Each line consists of three numbers separated by a space: the base of the number system, the least significant digit of the first factor, and the second factor. This second factor is one digit only hence less than the base. The input file ends with the standard end-of-file marker.

OUTPUT *(rotamult-output.txt)*

Your program determines for each input line the number of digits of the smallest first factor with the rotamultproperty. The output file is also a text file. Each line contains the answer for the corresponding input line.

SAMPLE RUN

Input	Output
10 7 4 9 7 4 17 14 12	6 2 4

Prepared by ***Mark Edward M. Gonzales***

Asian MathSci League, Inc (AMSLI)
Website: amsliphil.com
Email address: amsliphil@yahoo.com

PROBLEM 6: Maya Calendar

(1995 Central Europe ACM-ICPC)

During his last sabbatical, professor M. A. Ya made a surprising discovery about the old Maya calendar. From an old knotted message, professor discovered that the Maya civilization used a 365 day long year, called *Haab*, which had 19 months. Each of the first 18 months was 20 days long, and the names of the months were *pop, no, zip, zotz, tzec, xul, yoxkin, mol, chen, yax, zac, ceh, mac, kankin, muan, pax, koyab, cumhu*. Instead of having names, the days of the months were denoted by numbers starting from 0 to 19. The last month of Haab was called *uayet* and had 5 days denoted by numbers 0, 1, 2, 3, 4. The Maya believed that this month was unlucky, the court of justice was not in session, the trade stopped, people did not even sweep the floor.

For religious purposes, the Maya used another calendar in which the year was called *Tzolkin* (holly year). The year was divided into thirteen periods, each 20 days long. Each day was denoted by a pair consisting of a number and the name of the day. They used 20 names: *imix, ik, akbal, kan, chicchan, cimi, manik, lamat, muluk, ok, chuen, eb, ben, ix, mem, cib, caban, eznab, canac, ahau* and 13 numbers; both in cycles.

Notice that each day has an unambiguous description. For example, at the beginning of the year the days were described as follows:

> *1 imix, 2 ik, 3 akbal, 4 kan, 5 chicchan, 6 cimi, 7 manik, 8 lamat, 9 muluk, 10 ok, 11 chuen, 12 eb, 13 ben, 1 ix, 2 mem, 3 cib, 4 caban, 5 eznab, 6 canac, 7 ahau,* and again in the next period *8 imix, 9 ik, 10 akbal...*

Years (both Haab and Tzolkin) were denoted by numbers 0, 1, ..., where the number 0 was the beginning of the world. Thus, the first day was:

Haab: `0. pop 0`

Tzolkin: `1 imix 0`

Help professor M. A. Ya and write a program for him to convert the dates from the Haab calendar to the Tzolkin calendar.

INPUT *(maya-input.txt)*

The data in Haab is given in the following format:

NumberOfTheDay. Month Year

The first line of the input file contains the number of the input dates in the file. The next n lines contain n dates in the Haab calendar format, each in separate line. The year is smaller than 5000.

OUTPUT *(maya-output.txt)*

The date in Tzolkin should be in the following format:

Number NameOfTheDay Year

The first line of the output file contains the number of the output dates. In the next n lines, there are dates in the Tzolkin calendar format, in the order corresponding to the input dates.

SAMPLE RUN

Input	Output
3 10. zac 0 0. pop 0 10. zac 1995	3 3 chuen 0 1 imix 0 9 cimi 2801

PROBLEM 7 STARTS ON THE NEXT PAGE

Asian MathSci League, Inc (AMSLI)
Website: amsliphil.com
Email address: amsliphil@yahoo.com

PROBLEM 7: Repeating Decimals

(1990 World Finals ACM-ICPC)

The decimal expansion of the fraction $1/33$ is $0.00\overline{3}$, where the $\overline{03}$ is used to indicate the cycle 03 repeats indefinitely with no intervening digits. In fact, the decimal expression of every rational number (fraction) has a repeating cycle as opposed to decimal expansions of irrational numbers, which have no such repeating cycles. Examples of decimal expansions of rational numbers and their repeating cycles are shown below. Here, we use parentheses to enclose the repeating cycle rather than place a bar over the cycle.

Fraction	Decimal Expansion	Repeating Cycle	Cycle Length
1/6	0.1(6)	6	1
5/7	0.(714285)	714285	6
1/250	0.004(0)	0	1
300/31	9.(677419354838709)	677419354838709	15
655/990	0.6(61)	61	2

Write a program that reads numerators and denominators of fractions and determines their repeating cycles. For the purposes of this problem, define a repeating cycle of a fraction to be the first minimal length string of digits to the right of the decimal that repeats indefinitely with no intervening digits. Thus for example, the repeating cycle of the fraction 1/250 is 0, which begins at position 4 (as opposed to 0 which begins at positions 1 or 2 and as opposed to 00 which begins at positions 1 or 4).

INPUT *(decimals-input.txt)*

Each line of the input file consists of an integer numerator, which is nonnegative, followed by an integer denominator, which is positive. None of the input integers exceeds 3000. End-of-file indicates the end of input.

OUTPUT *(decimals-output.txt)*

For each line of input, print the fraction, its decimal expansion through the first occurrence of the cycle to the right of the decimal or 50 decimal places (whichever comes first), and the length of the entire repeating cycle. In writing the decimal expansion, enclose the repeating cycle in parentheses when possible. If the entire repeating cycle does not occur within the first 50 places, place a left parenthesis where the cycle begins — and place "**...**)" after the 50th digit.

SAMPLE RUN

Input	Output

76 25
5 43
1 397
	76/25 = 3.04(0)
	  1 = number of digits in repeating cycle
	
	5/43 = 0.(116279069767441860465)
	  21 = number of digits in repeating cycle
	
	1/397 = 0.(002518891687657430730478589420654911838790931989 92...)
	  99 = number of digits in repeating cycle

-- return 0; --