Eighth Annual Taylor University Programming Contest

Jonathan Geisler David Gallagher Jim Knisely Victor Norman
Brent Wilson

April 7, 2018

1 `frange`: a new Python function

1.1 Problem Description

You've been learning about Python and like the `range` function, but feel like it should be extended to support real numbers. You've decided to implement a companion to `range` called `frange` which acts the same except that it accepts real numbers instead of integers.

For those of you that do not have the Python documentation memorized, here is the relevant documentation from the online manual:

> `def range(start, stop, step):`
>
> For a positive `step`, the contents of a range `r` are determined by the formula `r[i] = start + step*i` where `i` ≥ 0 and `r[i]` $<$ `stop`.
>
> For a negative `step`, the contents of the range are still determined by the formula `r[i] = start + step*i`, but the constraints are `i` ≥ 0 and `r[i]` $>$ `stop`.
>
> A range object will be empty if `r[0]` does not meet the value constraint.

The last line means that if `r[0]` cannot be created, then it will generate an empty array.

1.2 Input

The first line of the input will be a single integer, n, which represents the total number of unit test cases to run. The next n lines will each contain three integers: b, e, and s. The value b should be used as the `start` value for your function, the value e should be used as the `stop` value for your function; the value s should be used as the `step` value for your function.

1.3 Output

For each triplet of values, b, e, and s, your program should output a line representing the array produced by your `frange` function. The array should start with a `[` character, be followed by the values in the array and end with a `]` character. The values in the array should be separated by a comma and space and be rounded to four decimal places.

1.4 Example

Input	Output
4 1.0 2.0 0.25 1.0 0.0 -0.2 1.0 2.0 5.0 1.0 0.0 2.5	[1.0000, 1.2500, 1.5000, 1.7500] [1.0000, 0.8000, 0.6000, 0.4000, 0.2000] [1.0000] []

2 Just add two numbers

2.1 Problem Description

I give you two numbers; you give me the sum. How hard could that be?

OK, maybe I'll give you the numbers in some base other than 10. And maybe the numbers will be real numbers instead of integers. How hard could that be?

If I give you two base-10 numbers like 13.2 and 3.43, you could give me the sum 16.63. But if I give you two base-2 numbers like 11.1 and 101.11, you should give me the sum of 1001.01. And if I asked for the base-10 equivalent of that sum, you could tell me 9.25. Piece of cake!

2.2 Input

The first line of input will be an integer n. The input set will contain n test cases. Each test case will be on a single line of input with three values: b, x, and y. The first integer of that line will be the base, b, of the numbers you will add ($2 \leq b \leq 36$). The remaining two positive numbers on the line (x and y) will be the numbers to be added, as represented in the given base. These numbers will always contain a dot (`.`) splitting the integer and fractional portion of the number. The integer portion of each number will contain at least one digit and no more than ten digits. Likewise, the fractional portion of each number will contain at least one digit and no more than ten digits. For bases larger than ten, the numbers will use uppercase letters `A-Z`, where `A` is the equivalent of the decimal 10; `Z` represents the decimal 35. Note that this is just a simple extension of how we represent base-16 (i.e., hexadecimal).

2.3 Output

For each test case, output a single line containing two sums, s_b and s_{10}, separated by a single space. s_b should provide an exact answer, but with no extraneous leading or trailing zeros. For s_b, if the fractional part of the sum is zero, don't print the dot or the fractional part. For s_{10}, the fractional portion should be rounded to exactly four fractional digits as shown in the Example Output.

2.4 Example

Input	Output
5 10 13.2 3.43 2 11.1 101.11 3 12.2 222.12 36 ZZ.1 A2.Z 5 0.2 0.2	16.63 16.6300 1001.01 9.2500 1012.02 32.2222 1A2 1658.0000 0.4 0.8000

3 Left pyramidal marquee

3.1 Problem Description

The El Nasr Automotive Manufacturing Company of Egypt has asked you to program the radio display unit in its new cars. As a nod to the origin country of the vehicles, they've asked you to build a new display method for song titles called the "left pyramidal marquee." For a normal left marquee, the title would be displayed on screen and if it is too big, the title would scroll to the left. The difference between the two is that the title doesn't start full on the screen, but appears to grow like a pyramid.

Each second, a new stage in the marquee is displayed. The first second will display the first character of the title centered in the middle of the screen. On the next second, the display will appear to grow on either side of the original character by lengthening the title by two and continuing to center the new string. This lengthening will continue to apply until either the title is fully shown or the display is full. If the display is full and the title is not fully shown, the marquee will appear to be a left marquee until the last character of the title is displayed.

For example, the title, "Walk like an Egyptian" would be displayed in a left pyramidal marquee for an 17 character display as:

Time	Display
1	`W`
2	`Wal`
3	`Walk`
4	`Walk li`
5	`Walk like`
6	`Walk like a`
7	`Walk like an`
8	`Walk like an Eg`
9	`Walk like an Egyp`
10	`alk like an Egypt`
11	`lk like an Egypti`
12	`k like an Egyptia`
13	`like an Egyptian`

After the full title is displayed, the process repeats until the song is completed. Out of curiosity, the company wonders what the last view of the display is when a song completes. Your task is to show the last output on the display given a song name and length in seconds.

3.2 Input

The first line of input is a number, n, which represents the number of song titles to examine. The next n lines will contain the number of seconds in the song (l), and a song title (t) separated by a space.

3.3 Output

For each l and t, your program should output a line with the contents of what will be shown on a 17 character display if each step in the left pyramidal marquee takes one second.

3.4 Example

Input	Output

`3`
`5 Walk like an Egyptian`
`100 When you believe`
`165 Take me back to Cairo`
	`    Walk like`
	`     When yo`
	`Take me back to C`

4 Make me a new pyramid

4.1 Problem Description

The Pharaoh has commissioned you to find the plot of land to build his next pyramid. His demands are simple: find the square plot of land to support his building the largest possible pyramid. You will be provided a record of which subplots are being used and which are available. You cannot repurpose any land already being used and must find a square of the largest size possible.

4.2 Input

The program will have one line of input with two numbers: m and n ($1 \leq m, n \leq 5000$). The record of subplots will be $m \times n$ in size. There will be m lines and each will contain n characters. The value of `1` will indicate the associated subplot is available for building and the value of `0` will indicate the associated subplot is unavailable for building.

4.3 Output

The program will output one line with the size of the largest possible square in the input that only contains `1`'s.

4.4 Example[1]

Input	Output

`10 10`
`1111100000`
`1111110000`
`0111110000`
`0111111000`
`0011111000`
`0011111100`
`0001111100`
`0001111110`
`0000111110`
	`4`

`4 7`
`1010101`
`1111111`
`1111111`
`0101010`
	`2`

`8 5`
`11111`
`11111`
`11111`
`11111`
`11111`
`11111`
`11111`
`11111`
	`5`

[1]Note, for the examples in this problem, each input and output represent separate runs of the executable.

5 Musical steps

5.1 Problem Description

Your uncle, the Egyptologist, has a fascinating theory about the inner passageways within a pyramid. He believes that they are inspired by music with conjunct motion. Basically, he believes that each physical step up represents a rise in pitch by a minor second (also known as a half step) or major second (also known as a full step). Similarly, each physical step down represents a fall in pitch by a minor second or a major second. For those who are not musically inclined, the following half steps exist in modern western music: A $\rightarrow$ A$\sharp$ $\rightarrow$ B $\rightarrow$ C $\rightarrow$ C$\sharp$ $\rightarrow$ D $\rightarrow$ D$\sharp$ $\rightarrow$ E $\rightarrow$ F $\rightarrow$ F$\sharp$ $\rightarrow$ G $\rightarrow$ G$\sharp$ $\rightarrow$ A. A full step is two consecutive half steps. We will ignore the fact that we don't know how ancient Egyptian music worked and assume it worked similarly.

Furthermore, he believes that at the end of some passageways, there are doors that can only be opened by playing the melody of the passageway. Unfortunately, your uncle doesn't know the starting pitch of any given path and so would have to listen to quite a number of possible melodies to determine if his theory is correct.

He has asked you to help by telling him the number of melodies he will need to listen to so that he can perform an exhaustive search.

5.2 Input

The input will start with a line with one number (s), which represents the number of passageways your uncle wishes to explore. The next s lines will each contain a single number l, which is the number of steps of the given passageway. Your uncle has assured you that $0 < l \leq 60$.

5.3 Output

For each l, your program should output one line containing the number of possible melodies he will have to examine to explore his theory for the given passage length.

5.4 Example

Input	Output
3 1 11 23	12 12288 50331648

6 Pisano periods

6.1 Problem Description

An interesting result in mathematics is the Pisano period. As many students know, the Fibonacci sequence is defined as

$$F(n) = F(n-1) + F(n-2)$$
$$F(0) = 0$$
$$F(1) = 1$$

The Pisano sequence is defined as

$$P(i, n) = F(n) \mod i$$

It turns out that the Pisano sequence is cyclical for any given i. For example,

$$P(3, n) = 0, 1, 1, 2, 0, 2, 2, 1, 0, 1, 1, 2, 0, 2, 2, 1, 0, 1, 1, 2, 0, 2, 2, 1, 0, \ldots$$

By examining the sequence, we can see that the cycle length of this sequence is 8.

Your job is to compute the cycle length, or Pisano period, of any given Pisano sequence.

6.2 Input

The first line of input will contain a single number, n, which indicates the number of cycles to compute. The next n lines will each contain a number $i, 2 \leq i < 10,000$.

6.3 Output

For each i, you must output a line with the cycle length of $P(i, n)$.

6.4 Example

Input	Output
5 2 3 9 13 100	3 8 24 28 300

7 Snakes and ladders

7.1 Problem Description

Your wonderful niece has brought you the game of snakes and ladders. She likes the game, but thinks the game sometimes runs a bit long. She would like you to tell her how likely a game will end after a certain number of turns. You've decided to do her one better and tell her the likelihood of being on any given square after a certain number of turns.

You are probably more familiar with game as Chutes and Ladders published by Milton Bradley in 1943. The game is played on a 10×10 grid with some squares connected to snakes (chutes) and others to ladders. On each turn, a player rolls a fair six-sided die to determine how far forward to move. If the player lands on a snake, the player will follow the snake down to its tail; if the player lands on a ladder, the player will follow the ladder to its top. The object of the game is to be the first player to reach the last square on the board.

For our game, we are only going to consider one player on the board and compute the probability of where the player will be on any given turn. The squares on the board are numbered 1 through 100 and at the start of the game, the player begins off the board. The number and locations of the snakes and ladders will be provided to you as part of the program input.

7.2 Input

The first line of input will be a single number, n, indicating the total number of snakes and ladders. The next n lines will contain two numbers, b and e. If a player lands on a square numbered b, the player will move to the square numbered e. Note that this representation handles both snakes and ladders since snakes are represented by $b > e$ and ladders are represented by $b < e$. The value of b and e will never be the same. In fact, the end of a snake or ladder can never be the start of another snake or ladder.

Each of the remaining lines will be a pair of numbers, t and s. The value of t represents the turn number and cannot be larger than 1000. The value of s represents the square number.

The last line of input will be a pair of zeros. There should be no corresponding output for this pair of zeros.

7.3 Output

For each pair of t and s, the program should output a line with a number representing the probability that the player is on square s after turn t. The probability should be displayed as a percentage and rounded to four decimal places.

7.4 Example

Input	Output
19 1 38 4 14 9 31 21 42 28 84 36 44 51 67 71 91 80 100 16 6 47 26 49 11 56 53 62 19 64 60 87 24 93 73 95 75 98 78 1 1 1 2 1 38 3 7 50 100 1000 100 0 0	0.0000 16.6667 16.6667 3.2407 75.8681 100.0000

8 Stacking blocks

8.1 Problem Description

Your child has many six-sided color wooden block cubes. Each block's side is one of four colors: white, green, red, or blue. She wants to know, if she picks four blocks at random, whether she can stack those four blocks, one on top of the other, in such a way that her stack has all four colors on each of the four long sides of the stack. Being the programmer that you are, you decide to write a program so that your child can put the sides of her chosen four blocks in to see if it is possible. However, the program will only tell her if it's possible–not give her the answer, because what fun would that be?

You will be given four blocks each with six sides. You will also be given the color of each side. For our purposes, side 1 is the opposite side of the block from side 5, side 2 is the opposite side of the block from side 4, and side 3 is the opposite side of the block from side 6.

For example, it is possible to stack this set of blocks one on top of another such that each of the four long sides of the stack has all four colors present

Side	Block 1	Block 2	Block 3	Block 4
1	B	G	B	G
2	G	R	R	W
3	W	W	W	W
4	G	R	W	B
5	R	R	R	R
6	B	B	G	G

but it is not possible to stack this set of blocks one on top of another such that each of the four long sides of the stack has all four colors present.

Side	Block 1	Block 2	Block 3	Block 4
1	R	W	R	B
2	R	G	R	W
3	R	W	R	B
4	R	G	R	W
5	R	W	R	B
6	R	G	R	W

8.2 Input

The first line of the input will be a number $n, 1 \leq n \leq 10$, which represents the number of test cases in the input. The next n lines will each contain 24 characters with no spaces separating the characters. The first character will represent side 1 of block 1, the second character will represent side 2 of block 1 ..., the seventh character will represent side 1 of block 2, the eighth character will represent side 2 of block 2, ..., and so on.

8.3 Output

For each test case, the program will output one line indicating whether the blocks can be arranged properly. If the blocks can be arranged as desired, the output should be `Yes` and if the block cannot be arranged properly, the output should be `No` .

8.4 Example

Input	Output
2 BGWGRBGRWRRBBRWWRGGWWBRG RRRRRRWGWGWGRRRRRRBWBWBW	Yes No

9 What height is your pyramid?

9.1 Problem Description

You did such a good job of finding a plot of land for the next pyramid[2] that the Pharaoh would like you to get some more information for him. As you know, the base of the pyramid will be a square. The builders of the pyramid like symmetry, so they have decided that each face of the pyramid will be an equilateral triangle. Given the plot size, the Pharaoh would like to know how tall his pyramid will be.

9.2 Input

Each line of input will be an integer length, l, of one side of the square base. The input will be completed when the size is `0`.

9.3 Output

For each non-zero input, there will be one line of output indicating the height of the given pyramid. The output will be rounded to three decimal places.

9.4 Example

Input	Output
7 22 97 0	4.950 15.556 68.589

[2]See Problem 4