CodeLM 2017

Intermediate Division

School Name

Your codelm.com password is: ___________________________

Network login information:

ssid = lmsdcode
Password=lmsdcode

Rules: You will have until 11 o'clock complete your division's five questions. All questions must be submitted through the website codeLM.com and have a runtime of less than 10 seconds. You will submit your code by copying and pasting the code from your IDE to each problem's textbox on the website and then hitting submit.

Starter code should be downloaded from codeLM.com. **Do not change the names of the classes in this starter code or method headers.** If you do your code will not compile.

Problems 1 and 2 are each worth 2 points; Problems 3 and 4 are each worth 3 points; Problem 5 is worth 4 points. In order to receive credit for these questions your code must produce the correct results for all test data. This test data will not be the same as the sample data that is provided with each question in this packet, so be sure to thoroughly test your code before you submit it. Incorrect submissions that compile will result in the loss of one point. If your submission does not compile you will not lose a point. If you believe that your code has incorrectly been marked as wrong you should inform a judge who will then have the question reviewed. You may make as many submissions for each question as you would like, although you will only receive credit for one correct answer per problem.

Important Things to Remember

- You must download the starter code from codeLM.com. This starter code contains code that will read in all values that will be necessary to solve each problem from standard input.
- Do not change the class names in the starter code. If you do your code will not compile.
- Your code should not print out anything except for the answer. Your answer must be formatted exactly like the sample data on codeML.com or it will be marked wrong.

Index:

page #	problem
3	**1: Recalled Hardware** (2 points)
4-5	**2: Newmerals** (2 points)
6	**3: Encryption/Decryption** (3 points)
7	**4: Delivery Drone** (3 points)
8	**5: Password Cracking** (4 points)

Problem 1: Recalled Hardware

New Wave Computer's™ new product, the Universe Edge, has had some minor "malfunctions". Specifically, their phones' batteries have been know to charge slowly and spontaneously combust. To avoid legal fallout, certain Universe Edge units have to be recalled. You are to design a program that receives a serial number and determines if the phone is faulty.

Input

Integer **`serialNumber`**: The serial number is nine digits long. The first two digits in this serial number represent the **factory code**, digits 3 and 4 represent the **month of manufacture**, digits 5 and 6 represent the **year of manufacture** and digits 7, 8 and 9 form the **unique identifier** for each phone.

Output

A boolean which is true if the phone should be recalled and is false otherwise. A phone is faulty, and should therefore be recalled if its serial number meets any of the following conditions:

- The phone was manufactured in West Chester (factory code *66*).
- The phone was manufactured from November 2015 to February 2016 in New Jersey (factory codes *09 - 17*).
- Phones manufactured in October of 2016 in Malvern (factory code *84*) are faulty unless the unique identifier is divisible by 9, but not by 27.

Sample Data

Input	Output	Explanation
`serialNumber` `= 660915123`	`true`	The phone was manufactured in West Chester and is therefore faulty.
`serialNumber` `= 110915327`	`false`	Even though the phone was manufactured in a New Jersey factory, it was not manufactured between November 2015 and February 2016.
`serialNumber` `= 111115893`	`true`	The phone was manufactured in New Jersey between November 2015 and February 2016.

Problem 2: Newmerals

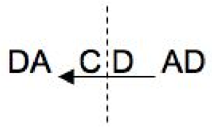
In order to improve security, New Wave Computers™ wants to implement a proprietary number system named Newmerals™. This system consists of four symbols: A, B, C, and D, where A < B, B < C and C < D. This system is so secretive that Newmerals cannot be translated into base-10 numbers. Newmerals can be combined using three operators: join, which is represented by a colon ':', promotion, which is represented by a carrot '^', and mirror, which is represented by a percent symbol '%'.

Input
Two Strings: **`newmeralA`** and **`newmeralB`**: Each string represents a two-digit Newmeral.

char **`operator`**: which is the operation that will be performed on the two Newmerals. This char can have one of three values: ':', '^' or '%'.

Output
A string that contains the Newmeral that will result from **`newmeralA`** combined with **`newmeralB`** using **`operator`**. Each of the three operations are described in the table below.

Operator	**Function**
Join ':'	Gather the two Newmerals by selecting the first digit from the first Newmeral, followed by the first digit from the second Newmeral, then the second digit from the first Newmeral and finally the second digit from the second Newmeral. This operation results in a four digit Newmeral. For example, **AB** : CD = **A**C**B**D (Digits are bolded and underlined for demonstration purposes only.)
Promotion '^'	Compare the two digits in each place value from the two Newmerals and construct a new Newmeral that is composed of the higher digit in each place value. For example, AB ^ CD = CD You would evaluate this expression by comparing the A from the first Newmeral with the C from the second Newmeral and selecting C. You would then compare the B from the first Newmeral to the D in the second Newmeral and select D. If two digits are the same, promote that digit by one (i.e. two C's would become D, etc.) If the two digits are both Ds, they should overflow and become an A. For example, AD ^ AD = BA In this example the two A's would be compared to each other and be promoted to B since they are the same. The two D's would also be compared to each other and

	be promoted to an A.
Mirror ‘%’	The Mirror operator reflects the second Newmeral over the first Newmeral. For example, CD % AD = DACDAD In this example AD is reflected over CD which would give you DACD.  AD is appended to the end of DACD to complete the operation.

Sample Data

Input	Output	Explanation
`newmeralA` = AC `newmeralB` = BD `operator` = :	ABCD	The ‘join’ operation is performed on the two Newmerals.
`newmeralA` = AD `newmeralB` = CD `operator` = ^	CA	The ‘promotion’ operation is performed on the two Newmerals. The operation is evaluated by comparing the A from the first Newmeral with the C from the second Newmeral and selecting C. You would then compare the D from the first Newmeral to the D in the second Newmeral. Since they are both the same Newmeral you would promote one, which in this case would wrap around to A.
`newmeralA` = AC `newmeralB` = DC `operator` = %	CDACDC	The reverse of **`newmeralB`**, “DC” is placed in front of **`newmeralA`** to get CDAC. **`newmeralB`** is then appended to the the end of this string.

Problem 3: nCrypt

New Wave has decided to encrypt all of their communications so that attackers cannot steal their trade secrets. They will use a highly advanced encryption technique in which each letter is replaced by the letter **x** letters after it in the alphabet. The message can include uppercase letters and lowercase letters. Any character that is not a letter should not be changed.

Using this algorithm, the word "The" encrypted with an offset of 4 would become "Xli", because 'X' is four letters after 'T' in the alphabet, 'l' is four letters after 'h' and 'i' is four letters after 'e'.

When decrypting a word a similar process is followed, only in the opposite direction. For example, "jUm" decrypted with an offset of 6 would result in "dOg".

If you were to attempt to encrypt a letter, but when counting **x** letters after that letter you find that you are outside of the alphabet, you should wrap around to the beginning of the alphabet. For example, if you were to encrypt the letter 'y' with an offset of 3, you would get the letter 'b'. The same would occur when decrypting, only you would wrap to the end of the alphabet.

Input
String **message**: The message that is to be either encrypted or decrypted.
int **x**: The distance that each letter will be offset.
char **operation**: This char will have one of two values; 'e' if **message** is to be encrypted or 'd' if **message** is to be decrypted.

Output
The encrypted or decrypted string.

Sample Data

Input	Output	Explanation
message = New Wave rocks **x** = 3 **operation** = e	Qhz Zdyh urfnv	Each letter in the the message is replaced by the letter three positions after it in the alphabet.
message = Rw. Xbtuj **x** = 5 **operation** = d	Mr. Swope	Each letter in the the message is replaced by the letter five positions before it in the alphabet. The 'b' in message becomes a 'w' because it will wrap around to the end of the alphabet. The period does not change.

Problem 4: Delivery Drone

New Wave Computers™ has decided to deploy a fleet of delivery drones. While their initial rollout has been promising, the drones have displayed an inability to scale some obstacles because of their rather weak batteries and because of their **inability to travel in direction other than toward the delivery point**. Luckily, battery packs have been scattered around strategic locations through the delivery zones. When making deliveries, a drone requires **n** battery packs to fly over **n** obstacles. In order for a drone to be able to make its delivery, it must have an available battery pack when it encounters an obstacle. This battery would then be discarded. A drone can carry multiple batteries at a time if they are available on its flight path.

You are to write a program that can determine if a drone will be able to make its delivery based on the number of obstacles and battery packs that it will encounter along its route.

Input
String `path`: A String that represents a drone's path. This string of unknown length will be composed from a collection of the following five characters:

- 'X' - The starting position of the drone. This starting point can be anywhere in the String. There will only be one starting point.
- 'O' - The delivery point. The delivery point can also be anywhere in the String. There will only be one delivery point.
- '-' - A clear path.
- '|' - An obstacle that must be flown over.
- '*' - A battery pack.

Output
A boolean which is true if the drone can make its delivery and is false otherwise.

Sample Data

Input	Output	Explanation
path = `X---*|--O`	true	There is one obstacle between the drone's starting and finishing points, however there is a battery pack that the drone can pick up before reaching this obstacle.
path = `O---*|---X`	false	There is an obstacle before there is a battery pack. Note that this path runs from right to left
path = `X--**|||*-O`	false	There are 3 obstacles and 3 battery packs but the drone would have only been able to pick up two battery packs before it encounters the third obstacle.
path = `*X---|--O`	false	The battery pack is not on the way to the obstacle and the drones can only travel towards the delivery point so it can not fly over the obstacle.

Problem 5: Password Cracking

An employee at New Wave Computers has stolen confidential documents and stored them on a laptop that is protected with a five-digit password. New Wave has several digits that they think they know, however others are missing all together. The range for each of these missing digits has been determined, as they do not always range from zero to nine. If an incorrect password is entered, the laptop will self-destruct, so you have been tasked with calculating the probability that the correct password can be guessed with the amount of information that is known.

Input

int **`guess`**: A five-digit integer. A zero in this integer represents an unknown digit.
int **`lower`**: The lower bound for any unknown number. The range is inclusive.
int **`upper`**: The upper bound for any unknown number. The range is inclusive.
int **`answer`**: The correct password. (Please don't ask why you would be guessing in the first place if you already know the correct password. You will be immediately disqualified if you do.)

Output

The probability of guessing the combination as an integer value. 10% would be returned as the integer 10. All values that would not resolve to a whole number should be **truncated** meaning you remove the decimal entirely(e.g. 8.9 -> 8).

Sample Data

Input	Output	Explanation
`guess` = 12021 **`lower`** = 1 **`upper`** = 9 **`answer`** = 12421	11	The range 1-9 inclusive contains 9 possible integers and there is only one digit that we are guessing, so the answer is 1/9 = 11.111...% which truncates to 11%
`guess` = 10401 **`lower`** = 1 **`upper`** = 5 **`answer`** = 12431	4	The range 1 - 5 contains 5 possible integers, and there are two digits must be guessed, so the chances are multiplied. ⅕ * ⅕ = 4%
`guess` = 12021 **`lower`** = 5 **`upper`** = 9 **`answer`** = 12421	0	The digit we're guessing for is not within the guess range, so there is a zero percent chance that the correct combination will be guessed.
`guess` = 54021 **`lower`** = 1 **`upper`** = 5 **`answer`** = 12345	0	There is a zero percent chance that the guess will match the actual password because the numbers that the employees thought they knew are actually wrong.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	