Frequently Asked Questions

How does the contest work?

To solve each problem, your team will need to write a computer program that reads input from the standard input channel and prints the expected output to the console. Each problem describes the format of the input and the expected format for the output. When you have finished your program, you will submit the source code for your program to the contest website. The website will compile and run your code, and you will be notified if your answer is correct or incorrect.

Who is judging our answers?

We have a team of Lockheed Martin employees responsible for judging the contest, however most of the judging is done automatically by the contest website. The contest website will compile and run your code, then compare your program's output to the expected official output. If the outputs match exactly, your team will be given credit for answering the problem correctly. Our judging team will review the website's work, but in most cases the automated response will stand.

How is each problem scored?

Each problem is assigned a point value based on the difficulty of the problem. When the website runs your program, it will compare your program's output to the expected judging output. If the outputs match exactly, you will be given the points for the problem. There is no partial credit; your outputs must match *exactly*. If you are being told your answer is incorrect and you are sure it's not, double check the formatting of your output, and make sure you don't have any trailing whitespace or other unexpected characters.

We don't understand the problem. How can we get help?

If you are having trouble understanding a problem, you can submit questions to the problems team through the contest website. While we cannot give hints about how to solve a problem, we may be able to clarify points that are unclear. If the problems team notices an error with a problem during the contest, we will send out a notification to all teams as soon as possible.

Our program works with the sample input/output, but it keeps getting marked as incorrect! Why?

Please note that the official inputs and outputs used to judge your answers are MUCH larger than the sample inputs and outputs provided to you. These inputs and outputs cover a wider range of test cases. The problem description will describe the limits of these inputs and outputs, but your program must be able to accept and handle any test case that falls within those limits. All inputs and outputs have been thoroughly tested by our problems team, and do not contain any invalid inputs.

We can't figure out why our answer is incorrect. What are we doing wrong?

Common errors may include:

- Incorrect formatting - Double check the sample output in the problem and make sure your program has the correct output format.
- Incorrect rounding - See the next section for information on rounding decimals.
- Invalid numbers - 0 (or 0.0, 0.00, etc.) is NOT a negative number. 0 may be an acceptable answer, but -0 is not.
- Extra characters - Make sure there is no extra whitespace at the end of each line of your output. Trailing spaces are not a part of any problem's output.
- Decimal format - We use the period (.) as the decimal mark for all numbers.

If these tips don't help, feel free to submit a question to the problems team through the contest website. We cannot give hints about how to solve problems, but may be able to provide more information about why your answers are being returned as incorrect.

I get an error when submitting my solution.

When submitting a solution, only select the source code for your program (depending on your language, this may include .java, .cs, .cpp, .py, or .vb files). Make sure to submit all files that are required to compile and run your program. Finally, make sure that the names of the files do not contain spaces or other non-alphanumeric characters (e.g. "Prob01.java" is ok, but "Prob 01.java" and "Bob'sSolution.java" are not).

Can I get answers to these problems after the contest?

Certainly! A member of our problems team will be available after the contest to answer any questions you have (and go over any incorrect answers you submitted so you can see why they were wrong). If you'd like a copy of our solutions to these problems, or those submitted by other teams, ask your coach to send an email to our Global Problems Lead, Brett Reynolds, at brett.w.reynolds@lmco.com. We can also provide copies of the official inputs and outputs used to judge your solutions.

How are ties broken?

At the end of the contest, teams will be ranked based on the number of points they earned from correct answers during the contest. If there is a tie for the top three positions in either division, ties will be broken as follows:

1. Fewest problems solved (this indicates more difficult problems were solved)
2. Fewest incorrect answers (this indicates they had fewer mistakes)
3. First team to submit their last correct response (this indicates they worked faster)

Please note that these tiebreaker methods may not be fully reflected on the contest website's live scoreboard. Additionally, the contest scoreboard will "freeze" 30 minutes before the end of the contest, so keep working as hard as you can!

Mathematical Information

Rounding

Some problems will ask you to round numbers. All problems use the "half up" method of rounding unless otherwise stated in the problem description. Most likely, this is the sort of rounding you learned in school, but some programming languages use different rounding methods by default. **Unless you are certain you know how your programming language handles rounding, we recommend writing your own code for rounding numbers based on the information provided in this section.**

With "half up" rounding, numbers are rounded to the nearest integer. For example:

- 1.49 rounds down to 1
- 1.51 rounds up to 2

The "half up" term means that when a number is exactly in the middle, it rounds to the number with the greatest absolute value (the one farthest from 0). For example:

- 1.5 rounds up to 2
- -1.5 rounds down to -2

Rounding errors are a common mistake; if a problem requires rounding and the contest website keeps saying your program is incorrect, double check the rounding!

Trigonometry

Some problems may require the use of trigonometric functions, which are summarized below. Most programming languages provide built-in functions for $\sin X$, $\cos X$, and $\tan X$; consult your language's documentation for full details. Unless otherwise stated in a problem description, it is *strongly recommended* that you use your language's built-in value for pi (π) whenever necessary.
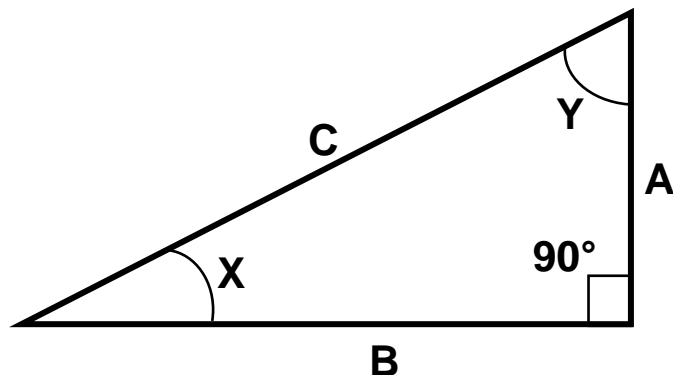
$$\sin X = \frac{A}{C}$$

$$\cos X = \frac{B}{C}$$

$$\tan X = \frac{A}{B}$$

$$X + Y = 90°$$

$$A^2 + B^2 = C^2$$

$$\frac{degrees * \pi}{180} = radians$$

US ASCII Table

The inputs for all Code Quest® problems make use of printable US ASCII characters. Non-printable or control characters will not be used in any problem unless explicitly noted otherwise within the problem description. In some cases, you may be asked to convert characters to or from their numeric equivalents, shown in the table below.

Binary	Decimal	Character	Binary	Decimal	Character	Binary	Decimal	Character
0100000	32	(space)	1000000	64	@	1100000	96	`
0100001	33	!	1000001	65	A	1100001	97	a
0100010	34	"	1000010	66	B	1100010	98	b
0100011	35	#	1000011	67	C	1100011	99	c
0100100	36	$	1000100	68	D	1100100	100	d
0100101	37	%	1000101	69	E	1100101	101	e
0100110	38	&	1000110	70	F	1100110	102	f
0100111	39	'	1000111	71	G	1100111	103	g
0101000	40	(	1001000	72	H	1101000	104	h
0101001	41	)	1001001	73	I	1101001	105	i
0101010	42	*	1001010	74	J	1101010	106	j
0101011	43	+	1001011	75	K	1101011	107	k
0101100	44	,	1001100	76	L	1101100	108	l
0101101	45	-	1001101	77	M	1101101	109	m
0101110	46	.	1001110	78	N	1101110	110	n
0101111	47	/	1001111	79	O	1101111	111	o
0110000	48	0	1010000	80	P	1110000	112	p
0110001	49	1	1010001	81	Q	1110001	113	q
0110010	50	2	1010010	82	R	1110010	114	r
0110011	51	3	1010011	83	S	1110011	115	s
0110100	52	4	1010100	84	T	1110100	116	t
0110101	53	5	1010101	85	U	1110101	117	u
0110110	54	6	1010110	86	V	1110110	118	v
0110111	55	7	1010111	87	W	1110111	119	w
0111000	56	8	1011000	88	X	1111000	120	x
0111001	57	9	1011001	89	Y	1111001	121	y
0111010	58	:	1011010	90	Z	1111010	122	z
0111011	59	;	1011011	91	[	1111011	123	{
0111100	60	<	1011100	92	\	1111100	124		
0111101	61	=	1011101	93	]	1111101	125	}
0111110	62	>	1011110	94	^	1111110	126	~
0111111	63	?	1011111	95	_			

Terminology

Throughout this packet, we will describe the inputs and outputs your programs will receive. To avoid confusion, certain terms will be used to define various properties of these inputs and outputs. These terms are defined below.

- An **integer** is any whole number; that is, a number with no decimal or fractional component: -5, 0, 5, and 123456789 are all integers.
- A **decimal number** is any number that is not an integer. These numbers will contain a decimal point and at least one digit after the decimal point. -1.52, 0.0, and 3.14159 are all decimal numbers.
- **Decimal places** refer to the number of digits in a decimal number following the decimal point. Unless otherwise specified in a problem description, decimal numbers may contain any number of decimal places greater or equal to 1.
- A **hexadecimal number** or **string** consists of a series of one or more characters including the digits 0-9 and/or the uppercase letters A, B, C, D, E, and/or F. Lowercase letters are not used for hexadecimal values in this contest.
- **Positive numbers** are those numbers strictly greater than 0. 1 is the smallest positive integer; 0.000000000001 is a very small positive decimal number.
- **Non-positive numbers** are all numbers that are not positive; that is, all numbers less than or equal to 0.
- **Negative numbers** are those numbers strictly less than 0. -1 is the greatest negative integer; -0.000000000001 is a very large negative decimal number.
- **Non-negative numbers** are all numbers that are not negative; that is, all numbers greater than or equal to 0.
- **Inclusive** indicates that the range defined by the given values includes both of the values given. For example, the range 1 to 3 inclusive contains the numbers 1, 2, and 3.
- **Exclusive** indicates that the range defined by the given values does not include either of the values given. For example, the range 0 to 4 exclusive includes the numbers 1, 2, and 3; 0 and 4 are not included.
- **Date and time formats** are expressed using letters in place of numbers:
  - **HH** indicates the hours, written with two digits (including a leading zero when needed). The problem description will specify if 12- or 24-hour formats should be used.
  - **MM** indicates the minutes for times or the month for dates. In both cases, the number is written with two digits (including a leading zero when needed). January is month 01.
  - **YY** or **YYYY** is the year, written with two or four digits (including a leading zero when needed).
  - **DD** is the date of the month, written with two digits (including a leading zero when needed).