Code Quest Contest Preparation

2021 Annual International Contest

Saturday, April 24, 2021

Table of Contents

Introduction

Welcome to Lockheed Martin's Code Quest competition! We're looking forward to meeting you.

Lockheed Martin's Code Quest competitions are international contests held every year for students in grades 6-12 (or their international equivalents). Founded in 2012 at the Fort Worth facility, Code Quest events now span the globe, with 27 sites across the United States, Canada, the United Kingdom, Poland, Australia, and Singapore. Our mission is to encourage young students to pursue their interests in computer programming and improve their skills in a competitive, yet fun and encouraging environment. We also provide teachers with resources to improve the quality of programming education.

Your goal during this contest is to try to complete the quest by earning as many points as possible. Points are earned by writing software programs in Java, Python, C#, C++, or Visual Basic in order to solve a series of problems. The contest only lasts for two and a half hours, so you and your team members will need to solve as many problems as possible within that time.

If you've participated in Code Quest events in the past, things will be a little different this year. Your team will connect to a Zoom conference call for the contest; once the contest begins, your team will be moved to a private breakout room within the Zoom call, allowing you to privately discuss your work with your team members. Since you'll all be at your own computer, you'll also be able to each work on a different problem at the same time! We'll be providing you with a list of 30 problems to make sure that you'll have plenty to work on during the contest.

Despite the situation, we're trying to keep things as normal as possible. If you run into trouble during the event, Lockheed Martin volunteers will be available to help you both on the Zoom call and by asking questions through the contest website. We'll also have some activities for you to participate in before the contest while we finish setting up, and afterward while we tally up the results.

The rest of this document will explain how the contest works and provide you with practice problems that you and your teammates can work on immediately. If you have any questions about this contest, please have your coach contact us at code-quest.gr-aero@lmco.com , and we'll do our best to answer as quickly as possible.

Good luck! We'll see you on April 24th!

Things To Do Now

In order to prepare for the contest and ensure your team can perform at their best, there are a few things we recommend you work on now.

Decide On a Strategy With Your Teammates

As mentioned above, this year's contest is going to be a little different. Rather than your entire team sitting around one computer in a crowded room, you'll each be at your own computer. You'll need to decide how best to work together to maximize your success.

- Identify your strengths and weaknesses - if you struggle with a particular algorithm or concept, one of your teammates might be able to give you some pointers.
- Determine how you'll pick problems to work on. Should you start with the easy problems and work up, or try to get an early lead by starting on a harder problem?
- Work out a strategy for solving problems - will you each work problems on your own, or work together on a single problem? What should you do if you get stuck on a problem?

Set Up Your Environment

If you're taking a programming class or using a school-provided computer, most likely you already have a development environment ready to go, but just in case, here are some development tools we recommend for use with Code Quest problems. This list includes installable software, which is usually more robust and provides more features, as well as some online development tools, which may be better suited to working collaboratively with your team, and may be necessary when working with school computers or Chromebooks.

- Installed IDEs:
  - Java/C++ - Eclipse IDE https://www.eclipse.org/downloads/
  - Python - Python IDLE (included with most Python installs) https://www.python.org/downloads/
  - C#/Visual Basic - Microsoft Visual Studio Community https://visualstudio.microsoft.com/vs/community/
- Online IDEs
  - Replit - https://repl.it/
  - Codeanywhere - https://codeanywhere.com/
- Text editing - Notepad++ https://notepad-plus-plus.org/downloads/
- File comparisons - Beyond Compare https://www.scootersoftware.com/

Legal stuff: Lockheed Martin is not affiliated with, and does not endorse, authorize, nor sponsor any of the websites above or their respective owners, companies, and products. If you decide to access any of these links or utilize the software listed above, you do so entirely at your own risk.

Solve Practice Problems

Once you have everything set up and ready to go, read through the remainder of this document to learn how the contest works and what will be expected of you and your teammates. At the end of the document you'll find Problems A and B, our practice problems. These problems will be available for you to solve on the day of the contest, prior to the actual start of the contest. Solutions for Problem A are included in this document, but you have to come up with a solution for Problem B on your own!

Get Your Credentials Ready

Your coach should have received a list of usernames and passwords for each member of your team in the same email that contained this document. Make sure you know what your username and password is - it's different for each team member! Your password will contain three lowercase letters, three uppercase numbers, two numbers, and one special character in a random order. In order to avoid confusion, certain characters will not appear in your password:

- The lowercase English letter 'l' (as in 'lemon')
- The uppercase English letter 'I' (as in 'Issac')
- The numeral 0 (zero)

If you have trouble logging into the contest website on the day of the event, volunteers will be able to provide you with your username and password, and will have the ability to reset your password if needed. On the day of the contest, we recommend logging in as early as possible to allow time to resolve any technical issues or answer any questions you may have.

Practice Old Problems

Once you've done all of the above, you're ready to go! If you'd like to try your hand at some other Code Quest problems, visit our new Code Quest Academy website at https://lmcodequestacademy.com . You can also download problems and solutions for problems from our contests from 2014 and earlier from our website at https://www.lockheedmartin.com/en-us/who-we-are/communities/codequest.html .

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.

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 any 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, .c, .h, .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 solutions to the problems after the contest?

Yes! Please ask your coach to email Code Quest® Problems Lead Brett Reynolds at brett.w.reynolds@lmco.com .

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

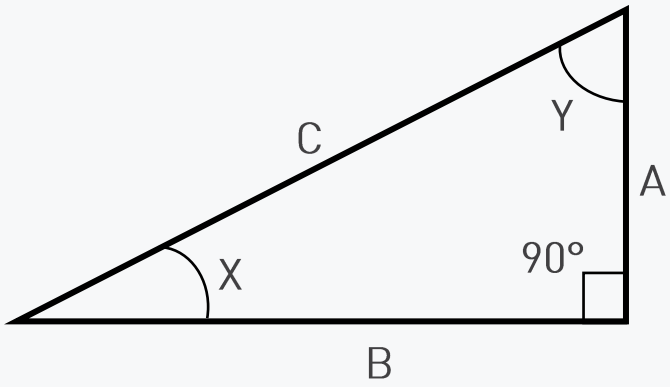
Some problems may require the use of trigonometric functions, which are summarized below. Most programming languages provide built-in functions for $\sin X$, $\cos X$, and $\tan X$; consult your language's documentation for full details. Unless otherwise stated in a problem description, it is *strongly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

$$\sin X = \frac{A}{C} \quad \cos X = \frac{B}{C} \quad \tan X = \frac{A}{B} = \frac{\sin X}{\cos X}$$

$$X + Y = 90°$$

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

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



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 (with a leading zero if 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 (with a leading zero if needed; January is 01).
    - **YY** or **YYYY** is the year, written with two or four digits (with a leading zero if needed).
    - **DD** is the date of the month, written with two digits (with a leading zero if needed).

Problem A: Hello, World!

Points: 0 (This is a practice problem)

Problem Background

Welcome to the 2021 Code Quest competition! We're excited to have you join us. Good luck today!

Before the contest begins, we want to make sure you are able to connect to and use the judging software. This doesn't require you to install or run any special software on your computer; all you need is a web browser, which should already be installed. You can connect to the contest website by going to the URL in the email your coach received prior to the contest, which also contains your personal username and password.

Once you're connected, continue to the next section to learn how to write and submit your solutions.

Problem Description

We want you to submit a sample program just to make sure everything is working correctly. Don't worry about writing the code; we give you the answers to this problem!

While sample input and output files will be provided to allow you to test your programs on your computers, your programs need to be able to read all input from the standard input channel, and print it to the standard output channel (the console). IDEs such as Eclipse and NetBeans can be configured to feed the contents of these files into your program's standard input. If you run your programs from the command line (in Windows, Mac, or Linux), you can pass the file's content into standard input like so, assuming the source and input files are in the current directory:

```
{command to compile your program}
{command to run your program} < {input file}
```

Example for C++:

```
g++ -o Prob05.exe Prob05.cpp
Prob05.exe < Prob05.in.txt
```

Example for Java:

```
javac HelloWorld.java
java HelloWorld < HelloWorld.in.txt
```

For this sample problem, all we want you to do is print out whatever input you receive. Again, the provided source code will do just that, but you're welcome to try to write your own solution

once you're sure everything is working. We'd also recommend you try your hand at solving Problem B for additional practice.

If you run into any problems now or during the contest, please let us know on the Zoom call or by submitting a clarification through the contest website. Good luck!

Sample Input

The first line of your program's input, **received from the standard input channel**, will contain a positive integer representing the number of test cases. Each test case will include a single line of text to be reprinted to the standard output channel.

```
2
Welcome to Code Quest!
Good luck today!
```

Sample Output

For each test case, your program must output the provided input line, unchanged.

```
Welcome to Code Quest!
Good luck today!
```

Solution Code

The solution code for each language, and specific details for working with each language, are provided below. **We strongly recommend using this code as a template for solving all other problems during the contest.** When submitting solutions, submit the source code only (e.g. .java, .cpp, .vb, .cs, or .py files); do not submit executable files (e.g. .exe files) or compiled code (e.g. .class files).

Java

DOMJudge supports Java 8. The use of 'package' statements is unnecessary, but will not cause compilation errors.

```java
import java.util.Scanner;

public class Prob00 {
  public static void main(String[] args) {
    try (Scanner input = new Scanner(System.in)){
      int testCases = Integer.parseInt(input.nextLine());

      for(int testcase = 0; testcase < testCases; testcase++) {
        System.out.println(input.nextLine());
      }
    }
  }
}
```

Python

DOMJudge supports Python 3.

```
# Recommended imports for all problems
# Some problems may require more
import sys
import math
import string

cases = int(sys.stdin.readline().rstrip())
for caseNum in range(cases):
  print(sys.stdin.readline().rstrip())
```

C#

DOMJudge supports .NET version 4.6 and C# version 6.0.

```
using System;

class CodeQuest {
  static void Main(string[] args) {
    int numTestCases = Convert.ToInt32(Console.ReadLine());

    for(int testCase = 0; testCase < numTestCases; testCase = testCase + 1){
      string text = Console.ReadLine();
      Console.WriteLine(text);
    }
  }
}
```

C++

DOMJudge uses version 7.3.0 of the g++ compiler.

```
// Recommended includes for all problems. Some problems require additional
libraries.
#include <iostream>
#include <string>
#include <cmath>
#include <cstdlib>
using namespace std;

int main()
{
  int testCases;
  cin >> testCases;
  string dummy;
  getline(cin, dummy);

  for(int testcase = 0; testcase < testCases; testcase++){
    string text;
    getline(cin, text);
    cout << text << '\n';
  }
```

```
}
```

VB.NET

DOMJudge supports .NET version 4.6 and Visual Basic 8.

```
Option Explicit On

Module HelloWorld

    Sub Main(args As String())
        Dim i As Integer
        Int32.TryParse(Console.ReadLine(), i)
        Dim test As Integer = 0
        For test = 0 To (i - 1)
            Console.WriteLine(Console.ReadLine())
        Next
    End Sub

End Module
```

Problem B: Not So Self-Driving

Points: 0 (This is a practice problem)

Author: Chris Liu, California High School

Special Acknowledgement

Unlike all other problems you'll see in the contest, this problem was not written by a Lockheed Martin employee. Chris Liu is a senior at California High School who attended the Code Quest competition in Sunnyvale, California last year. Since then, he has created two of his own programming competitions modeled on the Code Quest competition. Chris is also the lead director for the SRC Hacks hackathon held in February 2019 in San Ramon, California, which was attended by hundreds of other high school students. The problem below was written by Chris for a programming competition at SRC Hacks, and is presented here with his permission, modified only as needed to fit within our format.

Problem Background

Dwayne and Johnson are your partners in a group project for your autonomous-vehicles engineering course. However, none of you were paying attention in class, so you forget about the project until the night before it's due! Knowing there would be no time to train a decent self-driving algorithm in just a few hours, you, Dwayne, and Johnson all decide to fake a self-driving car with a few sensors and a bunch of if-statements.

Problem Description

Your task is to create an obstacle-avoidance system that will output instructions based on information about obstacle distance and the car's speed.

Sample Input

The first line of your program's input, **received from the standard input channel**, will contain a positive integer representing the number of test cases. Each test case will include a single line containing two numbers separated by a colon:

- A decimal value **V**, between 0 and 200 inclusive, denoting the vehicle's current speed (in m/s)
- A decimal value **X**, between 1 and 400 inclusive, denoting the obstacle's distance from the front of the car (in m)

Both values will be rounded to two decimal places.

```
5
23.15:98.34
2.40:17.33
6.79:5.01
0.00:1.53
113.56:113.56
```

Sample Output

For each test case, your program must print instructions to the car based on the following criteria:

- If the car is projected to collide with the obstacle in one second or less at the current speed, your program should output SWERVE.
- Otherwise, if the car is projected to collide with the obstacle in five seconds or less at the current speed, your program should output BRAKE.
- Otherwise, your program should output SAFE.

```
BRAKE
SAFE
SWERVE
SAFE
SWERVE
```