IEEE SoutheastCon 2023 Software Competition Hints and Examples

15 April 2023

Welcome to the IEEE SoutheastCon 2023 Student Software Competition! The contest committee has worked to create a contest that we hope will be challenging but fun for all students who are involved.

The format of the competition is somewhat similar to that of the ICPC (International Collegiate Programming Contest, originally sponsored by ACM and Upsilon Pi Epsilon). This is a very popular form of software competition around the world, and this format is used by many online competitive programming platforms, such as CodeChef, CodeForces, HackerRank, and Kattis.

A few other online competitive programming sites, like LeetCode and TopCoder, and corporate competitions such as the Meta Hacker Cup and Google Code Jam, also have strong similarities but critical differences in their formats.

Whether you are familiar with those competitions or sites should not matter. This competition will also have unique differences in its format and rules.

We strongly recommend that you read through the rules carefully, and in their entirety, and bring a copy with you to the competition.

This document is intended to provide a number of specific hints and examples which may clarify some of the questions you might have about the competition. While this document makes references to the rules, and may attempt in a few ways to clarify them, it is not a normative addendum to the rules.

If you find a discrepancy or need clarification at the event, please contact the contest committee or judges. If before the event, please contact Stephen Hopkins, stephen.hopkins@ieee.org.

In case you need to skip around in this document for a particular language, here's an overview of the contents.

Judging Environment **3**
Language Compilers/Runtimes for Judging **3**
Installing Python 3
Installing Java 3
Installing C and C++ 3
A Reference Model for Offline Development Environment **4**
Installing VS Code 4
Using VS Code with Python 4
Using VS Code with Java 5
Using VS Code with C/C++ 5
Working Offline with VS Code 5
Offline Reference Materials **6**
Downloadable Python Language Reference 6
Downloadable Java Language Reference 6
Downloadable C/C++ Language Reference 6
Downloadable Multi-Language Reference 6
Helpful Code Samples 6
Off-Laptop Resources **7**
Example Problem: Keeping Positive! **8**
General Solution Guidelines **9**
Python 3 Solution to Example Problem **10**
Java 17 Solution to Example Problem **11**
C++17 Solution to Example Problem: **12**
C17 Solution to Example Problem: **13**
Judging and Testing Recommendations **14**
Development Environment Testing **15**
Command Line Testing – General **15**
Command Line Testing – Python 16
Command Line Testing – Java 16
Command Line Testing – C++ 16
Command Line Testing – C 16
Verifying Output 16
Printing and Submitting **16**

Judging Environment

Your code will be evaluated by the judges on a Windows 10 laptop with the following specifications:

- Windows version: 10 22H2 (19045.2604)
- CPU: Intel x86_64 Core i5-8350U @ 1.7GHz, 4 cores
- Memory: 8.0 GB

Information detailing how submitted programs will be judged is provided in a later section.

Language Compilers/Runtimes for Judging

The judge machines will have all 4 languages installed,, so that they can be used to evaluate submitted programs in all languages.

Installing Python

To get the exact version of Python used by the judges on Windows, install Python 3.10 from the Microsoft store. The current version is 3.10.10.

If you are using a Mac or Linux environment, you should be able to get Python 3.10 for your platform, but for this competition, any Python 3.7 or greater should be adequate for development.

Installing Java

To get the exact version of Java used by the judges on Windows, install using the MSI (Microsoft Installer) package of the Java 17 JDK from the Oracle website:
https://download.oracle.com/java/17/archive/jdk-17.0.6_windows-x64_bin.msi
The current release is 17.0.6.

If you are using a Mac or Linux environment, you should also be able to get Java 17 from this page:
https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

For this competition, any Java SE JDK version 8 or greater should be adequate for development.

Installing C and C++

To get the exact version of C and C++ Gnu compilers used by the judges, you must first install MSYS2 on Windows following the instructions at https://www.msys2.org/. The judges use the default installation paths. The specific downloaded installer used for the judge machines is here:
https://github.com/msys2/msys2-installer/releases/download/2023-03-18/msys2-x86_64-20230318.exe

Once MSYS2 is installed it will launch a Unix-like shell window. Continue following the instructions to install **gcc** using **pacman** with the command:

```
pacman -S mingw-w64-ucrt-x86_64-gcc
```

You may **exit** the shell window after installing gcc, unless you decide to install additional items using pacman—such as a debugger (mentioned later in this document).

Finally, you should update your Windows **PATH** to allow gcc, g++, and other Gnu tools to run from the Windows command prompt. Instructions for doing this are in the VS Code section below; or see step #6 on the page https://code.visualstudio.com/docs/cpp/config-mingw.

The current GCC release is 12.2. Whether you are on Windows or on a Mac or Linux system, if you have GCC version 11.2 or higher, build with C17 and C++17 support, then it should be adequate for development in this competition.

A Reference Model for Offline Development Environment

If you are used to coding in an online editor such as Replit or IDE | GeeksforGeeks or Programiz (or many others) then a "no internet" contest might seem to be a particular challenge to you. You will need to install a local development tool onto your laptop, one that can function without internet access.

This section explains how to set up Microsoft Visual Studio Code (VS Code) on Windows, which is an IDE (integrated development environment) that can be used with any or all of the 4 supported competition languages.

You are not required to use VS Code for the competition, and the judges will not build or run your code using VS Code for scoring. This is provided as a reference model; it is simply one way a development environment has been proven to work for an offline contest.

If you follow this guide to prepare your laptop, and your team has a backup laptop, don't forget to prepare (and test) the backup laptop in the same way.

Installing VS Code

The simplest way to install VS Code in Windows is simply to search for "Visual Studio Code" in the Microsoft Store and install it from there.

VS Code is also available for other platforms, including Mac and Linux. Download the relevant version from https://code.visualstudio.com/download and check the online documents for instructions.

Once VS Code is installed, run it and choose a project folder for your work. Refer to VS Code documents to understand "Trust" settings for folders.

Using VS Code with Python

After you have installed VS Code and created a project folder, keep your internet connected while you setup Python. From the **File** menu, select **New** and then name your new file something like **hello.py**.

VS Code will notice the ".py" extension and suggest that you load Python extensions. Once you do this, VS Code will be able to provide syntax coloring and auto-completion in Python source files, as well as run and debug.

When you install Python for Windows, it comes with its own development environment, called IDLE. If your team will use only Python, you might find IDLE to be sufficient for your work.

Using VS Code with Java

After you have installed VS Code and created a project folder, keep your internet connected while you setup Python. From the **File** menu, select **New** and then name your new file something like **hello.java**. VS Code will notice the ".java" extension and suggest that you load Python extensions. Once you do this, VS Code will be able to provide syntax coloring and auto-completion in Python source files, as well as run and debug.

There are many other Java development environments, but some of them are not very good at creating simple, single source file Java programs.

Using VS Code with C/C++

The best reference for using VS Code in Windows with C and/or C++ is here:
https://code.visualstudio.com/docs/cpp/config-mingw

Many of the critical steps described on that page are already covered previously in this document, but don't forget this extra **pacman** command from step #5, which enables debugging:

```
pacman -S --needed base-devel mingw-w64-x86_64-toolchain
```

Also be sure to edit the **Path** variable in your Windows Settings as described in step #6.

If you are using VS Code on a Mac or Linux system, you might already have **gcc** tools available and in your path. From the **File** menu, select **New** and then name your new file something like **hello.c** or **hello.cpp**. VS Code will notice the ".c" or ".cpp" extension and suggest that you load C/C++ extensions. Once you do this, VS Code will be able to provide syntax coloring and auto-completion in C and C++ source files, as well as run and debug.

Working Offline with VS Code

Once you have downloaded all the extensions you might need in VS Code, try disconnecting your internet and use VS Code to create a few simple programs.

If that goes well, you should be able to get through the Example Problem below (and the competition) without any internet connection.

Of course, you might also want some reference materials.

Offline Reference Materials

Oh, no! You can't reach Stack Overflow! Modern software development relies heavily on internet resources, including online language and library references, standards and specifications, and question/answer sites.

It's worth spending some time to think about what references you will need during the contest, and how best to prepare them in advance for offline use. Be sure to consider both on-laptop and off-laptop resources.

Downloadable Python Language Reference

Almost all of the standard Python documentation, including tutorials and standard library documentation, is available for offline use with a web browser. Download Python 3.10 documentation here:
https://docs.python.org/3.10/download.html

Downloadable Java Language Reference

Download Java JDK 17 documentation here:
https://www.oracle.com/java/technologies/javase-jdk17-doc-downloads.html

Downloadable C/C++ Language Reference

https://en.cppreference.com/w/Cppreference:Archives

Downloadable Multi-Language Reference

https://devdocs.io/

This site uses local web storage to save your preferred documentation. Available languages include Python 3.10, OpenJDK 17, C++, and C. Select 'Preferences', enable the languages you want, disable the ones you don't want. Due to reliance on cached web content, this might not be as reliable offline.

Helpful Code Samples

In addition to language references, it might be worthwhile to find specific examples of code in your preferred language that demonstrate the following, and save copies to your laptop:

- How the "standard input" stream works in your language(s)
- How to read from the standard input stream
- How to read line by line from an input stream
- How to detect end of file in an input stream
- Parsing integers from a line of text
- Parsing words or characters from a line of text

Some of these topics are covered (albeit lightly) in the example solutions below. (Try them and save them!) For others, you may be able to download github samples from relevant blog articles.

Whatever digital resources you prepare on your laptop, if your team has a backup laptop, don't forget to prepare it in the same way.

Off-Laptop Resources

Books, code printouts, and other non-machine-readable media are permitted as reference materials in the competition. E-readers and tablets are not permitted.

While it might seem useful to bring a giant stack of Computer Science textbooks covering all sorts of algorithms and data structures, consider a couple of factors.

First, if you aren't already familiar with the content of the books, you won't have much time to search through them; it's better to bring books that you know well enough to find specific topics quickly. A well-indexed language reference book might save your team time, if you can look up something while a teammate is typing or testing another program.

Second, you don't want to bring so many books that you don't have enough space in your competition area to put them. (Of course, we intend that each team will have ample space for 3 people to work together, but there's not likely enough room for a wheelbarrow full of books.)

For printouts, whether they are printouts of helpful code samples or language reference, please note from the rules that they should be in a binder or folder (no loose-leaf paper).

Similarly, you may also bring bound notebooks of blank paper. If so, don't forget your pencils/pens.

Example Problem: Keeping Positive!
(0 points)

Time limit for each test run: 1s

Submit *one* source file:
problem0.c, *or*
problem0.cpp, *or*
problem0.java, *or*
problem0.py

Write a program that accepts a list of integers as input, and outputs how many positive numbers were provided.

The input for this program will be a single line of no more than 200 characters. Within the line will be one or more integer values ranging from -500 to 500. Each integer will be separated from the next by a single space.

Output a single line containing only the number of positive integers that were in the input.

Example #1

Input:

```
1 2 3 -4 5 6
```

Output:

```
5
```

Explanation: The input numbers 1, 2, 3, 5, and 6 are positive integers. so that's a total of 5 positive integers.

Example #2

Input:

```
44 0 -7 123 -399
```

Output:

```
2
```

Explanation: The input numbers 44 and 123 are positive integers, so that's a total of 2 positive integers.

General Solution Guidelines

Regardless of your chosen language, make sure that your solution meets all of these criteria:

1. **Everything is in one source file for judges to compile and run.** Do not depend on any non-standard header files or non-standard libraries, or require a Makefile.
    - Note: C programs will be linked with the Gnu standard math library.
2. **The source file is named correctly**. The automated scoring system used by the judges will look only for the exact names in the problem specification.
3. **The program reads all input from the "standard input" stream**, which is provided by the operating system and the language. Consult your language references, or the language-specific examples below, for hints about how to do this.
4. **The program does not hardcode any filename for input or output.** Using standard input and standard output makes this unnecessary.
5. **The program does not depend on specific command-line arguments** in order to run correctly. The judging system will run the program without any parameters.
6. **The program does not display any prompts for input** (such as "Enter numbers here:"). Prompts will be considered extra output, and an automated scoring system won't be able to tell them apart from rest of your program's output.
7. **The program does not open windows on the desktop**, or require any graphical environment. Graphical window environments are not necessary to solve any of the problems.
8. **The program does not open sockets nor attempt to use the network**.
9. **The program does not probe or manipulate the environment in which it is being run**, that is, it does not attempt to search folders or files on the judge machine, it does not probe devices, disks, etc., it does not create sockets, launch other programs, access drivers, etc. A program which does these sorts of things would be considered disruptive to the contest.

Your program *may* do the following things, if you really think they are needed:

- Write "debug" outputs to the "standard error" stream (not the "standard output" stream). Consult your language documentation for how to do this. The standard error stream will be ignored during judging; however, be advised that excessive output to the standard error stream might slow down your program's execution time.
- Use multi-threading, if it is available in standard libraries for your chosen language. Note that this is not likely to make a big difference in your program's overall execution time.
- Use as much memory as you think is needed. The judges believe that none of the solutions will require huge amounts of memory. Excessive use of memory may cause your program to crash and receive no credit.

Specific solutions to the Example Problem follow.

Python 3 Solution to Example Problem

Here are 2 versions of a Python 3 program which solves the example problem. In either case, the source code must be saved in a file named "problem0.py" when it is submitted.

First is a short version with just the code:

```
list_of_nums = map(int, input().split())
total = 0
for n in list_of_nums:
    if n > 0:
        total += 1
print(total)
```

Here is a longer version with some explanatory comments:

```
# Use 'input()' to get an entire input line from the "standard input" stream.
# Do _not_ pass any kind of parameter to prompt the user for input, since that
# would cause extra output from this program, and confuse the scoring software.
line = input()

# Use 'split()' to break the input line into a list of individual strings,
# based on the blank spaces.
list_of_strings = line.split()

# Use 'map()' with 'int' to convert the list of strings into a list of integers.
list_of_nums = map(int, list_of_strings)

# Count the positive integers in the list of integers.
total = 0
for n in list_of_nums:
    if n > 0:
        total += 1

# Output the value to the "standard output" stream.
print(total)
```

Java 17 Solution to Example Problem

Here are 2 versions of a Java program which solves the example problem. The source code must be saved into a file named "problem0.java" when it is submitted.

First, a short version with just the code:

```
import java.util.*;
class problem0 {
	public static void main(String[] unused) throws Exception {
		Scanner sc = new Scanner(System.in);
		int total = 0;
		while (sc.hasNextInt()) {
			int n = sc.nextInt();
			if (n > 0) ++total;
		}
		System.out.println(total);
		sc.close();
	}
}
```

Here is a longer version of the same program with some explanatory comments:

```
// Note that there is no 'package' statement here!
// We must use the "default package" in order to compile a standalone source file.

import java.util.*;  // for Scanner

// We must use the class name that matches the required filename.
class problem0 {
	// The main class must have a main method, but we won't need the args.
	public static void main(String[] unused) throws Exception {

		// The "standard input" stream for Java is 'System.in'.
		// The 'Scanner' class will read space-separated elements from the input.
		Scanner sc = new Scanner(System.in);
		int total = 0;

		// Read and evaluate each integer from the input using 'nextInt()'
		// until there are no more, which we will know from 'hasNextInt()'.
		while (sc.hasNextInt()) {
			int n = sc.nextInt();
			if (n > 0) ++total;
		}
		// The "standard output" stream for Java is 'System.out'.
		System.out.println(total);
		sc.close();
	}
}
```

C++17 Solution to Example Problem:

Here are 2 versions of a C++17 program (for use with Gnu Compilers) which solves the example problem. The source code must be saved into a file named "problem0.cpp" when it is submitted.

First, a short version with just the code:

```
#include <bits/stdc++.h>
int main() {
	int n, total = 0;
	while (true) {
		std::cin >> n;
		if (std::cin.eof()) break;
		if (n > 0) ++total;
	}
	std::cout << total << std::endl;
	return 0;
}
```

Here is a longer version of the same program with some explanatory comments:

```
#include <bits/stdc++.h>

// A 'main' function is required, but we won't be using args.
int main() {
	int n, total = 0;

	// The "standard input" stream for C++ is 'std::cin'.
	// Loop to read and evaluate integers until the input stream
	// ends, which we detect via 'eof()', the end of file method.
	while (true) {
		std::cin >> n;
		if (std::cin.eof()) break;
		if (n > 0) ++total;
	}

	// The "standard output" stream for C++ is 'std::cout'.
	std::cout << total << std::endl;

	// Always return zero from main().
	return 0;
}
```

C17 Solution to Example Problem:

Here are 2 versions of a C17 program (for use with Gnu Compilers) which solves the example problem. The source code must be saved into a file named "problem0.c" when it is submitted.

First, a short version with just the code:

```c
#include <stdio.h>
int main() {
	int n, total = 0, result = 1;
	while (result) {
		result = scanf("%d", &n);
		if (result == 0 || result == EOF) break;
		if (n > 0) ++total;
	}
	printf("%d\n", total);
	return 0;
}
```

Here is a longer version of the same program with some explanatory comments:

```c
#include <stdio.h>

// A 'main' function is required, but we won't be using args.
int main() {
	int n, total = 0, result = 1;

	// Loop to read and evaluate the input numbers.
	while (result) {

		// The 'scanf' function reads the "standard input" stream for C.
		// The "%d" format parameter tells scanf to parse the next integer.
		result = scanf("%d", &n);

		// The result of scanf will be EOF (end of file) when the input
		// stream ends, or will be zero if an integer was not found.
		if (result == 0 || result == EOF) break;

		if (n > 0) ++total;
	}

	// The 'printf' function writes to the "standard output" stream for C.
	printf("%d\n", total);
	return 0;
}
```

Judging and Testing Recommendations

The example problem, "Keeping Positive!", provides 2 example inputs, each with the corresponding correct output. For convenience, repeatability, and to make sure we can run each test in the required time (1 second!), the judges recommend putting each input into a separate text file. Also, this is how the judges will do it.

It doesn't matter much what you name the file, because the program reads from the "standard input" stream, regardless of the language you have chosen, but it would be helpful to use a naming convention.

Filename: **input-0-example-1.txt**

```
1 2 3 -4 5 6
```

Filename: **input-0-example-2.txt**

```
44 0 -7 123 -399
```

Testing with the examples is a good idea, and we'll get to that below, but it's also a good idea to consider situations that might possibly change the behavior of your code, and come up with your own input cases.

For the example problem, here are a few more suggested inputs:

Filename: **input-0-test-3.txt**

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1
```

Filename: **input-0-test-4.txt**

```
-500
```

It's good to sure your program will be *thoroughly* tested, since you will get only one chance to submit it!

Development Environment Testing

When you run your program in a desktop development environment such as VS Code, it will create an window for your program, possibly within the environment itself.

VS Code on Windows will create two integrated Terminal windows (each running PowerShell by default), first one for the build and then one to run and debug the program.

You could type all the input in this window, one key at a time, but that might be awkward and error-prone. It is simpler to open the input text file, use "select all" and "copy" there, then switch back to that program output window and "paste" the input all at once.

For VS Code, click to select the run/debug Terminal window (the cursor may change from hollow to solid) and then use **<shift>+<insert>** to paste. Whether you typed the input or pasted it, you may also need to type **<ctrl>+Z** and then **<Enter>** (or, in Mac or Linux, **<ctrl>+D**) to end the input stream.

Your program's output should display in the same window. Depending on your program, your choice of language, and library routines, the output and input might be interleaved.

While testing in a desktop development environment works for quick test runs on your own, please note that the judges will not use VS Code or any similar tool to run your code. They will do all official testing at the command line.

Command Line Testing – General

The judges will run your submission using *redirection* on the command line to send an entire input file to your program. You may find it useful to do your own test runs following the same method as the judges.

To get to the command line on Windows, press **<Windows>+R**, then type "**cmd**" and press **<Enter>**. On a Mac or Linux system, open a Terminal window. An development tool like VS Code might give you a workable command line within the app. Be sure you know the basics of working with a command line.

You may need to modify your **PATH** environment variable in order to run commands at the command line. Check your programming language installation instructions and operating system documentation for details. You may also need to change the command line session's current working directory to get to the folder where your files are, using a command like "**cd myproject**" for example. Again, check your operating system command line reference.

Assuming that your input files (named as given above) and your solution source file (problem0.c, problem0.cpp, problem0.java, or problem0.py) are in the current working directory, here are examples of how to test your code at the command line, using the Example #1 input:

Command Line Testing – Python

```
python3 problem0.py < input-0-example-1.txt > output.txt
```

Command Line Testing – Java

```
javac problem0.java
java problem0 < input-0-example-1.txt > output.txt
```

Command Line Testing – C++

The judges will compile C++ code with the C++17 standard option enabled.

```
g++ --std=c++17 -o problem0 problem0.cpp
problem0 < input-0-example-1.txt > output.txt
```

Note: Mac, Linux, and similar command line environments would use `./problem0` on the second line.

Command Line Testing – C

The judges will compile C code with the C17 standard option, and link the standard math library.

```
gcc -lm --std=c++17 -o problem0 problem0.c
problem0 < input-0-example-1.txt > output.txt
```

Note: Mac, Linux, and similar command line environments would use `./problem0` on the second line.

Verifying Output

Once your program has created an **output.txt** file for a given input (or whatever name you chose for the output file), you can open that file in an editor to compare it to the correct/expected result.

The judges will likely use automation to test and score your program, rather than doing each test run manually. The input and output filenames will not be disclosed, but the automation will use command line redirection in the format shown above.

Printing and Submitting

When you need a file printed, remember to delete everything else on the USB drive. Only one file will be printed at a time, and you might have to wait a few minutes if the wrong one got printed.

When you submit your programs at the end of the contest, remember to copy only one file for each problem to the USB, and make sure any other files are deleted. If you have been working on a problem in multiple languages, you must choose only one language for the submission.

You don't have to submit code for every problem.