How To Compile Programs on the Command Line

In various integrated development environments, there is usually a button to click or a pull down menu to compile a program. For C++ and Java, here is how you can compile on the command line (Command Prompt in Windows or Terminal on a Mac):

```
>g++ -o programname.exe programname.cpp
>javac programname.java
```

If successful, there should be no output on the screen and either programname.exe (C++) is created or programname.class (Java) is created.

How to Test Programs

Write your program and for C++ or Java, compile that program. In C++ a .exe file is created and in Java a .class file is created. In Python, no compilation step is needed since Python is fully interpreted.

To test a program on your own, please type up some test cases in a file first and give that file a name. (I typically call my files problemname.in, where problemname is the name of the problem.) Place this file in the same directory as your .exe file (for C++), .class file (for Java) or .py file (for Python).

Open up Command Prompt (in Windows) or Terminal (Mac). Change directories to where both the file to execute/interpret and test input file are. (cd is the command to change directories.)

Then, on the command line, do one of the following:

```
>programname.exe < testfile.in
>java programname < testfile.in
>python programname.py < testfile.in
```

When you do this, the output should come in the terminal window. If you want the output to be written in a file instead, do the following:

```
>programname.exe < testfile.in > myanswers.out
>java programname < testfile.in > myanswers.out
>python programname.py < testfile.in > myanswers.out
```

Now, when you do this, there will be no output to the screen as it's been redirected to the file myanswers.out. When you do this, make sure no meaningful file named myanswers.out exists. If it does, this will overwrite those contents.

Now, if you know what the correct answers are and store them in testfile.out, you can see if your answers match by doing this on the command line:

```
>fc myanswers.out testfile.out
```

If the files have the same exact contents, then this will produce the result:

```
fc: No differences found.
```

or something to that effect.

If they don't fc produces some output that is hard to follow. In these cases, you may want to use a tool like WinMerge, which does a better job of highlighting where two files are different. You can download WinMerge here:

https://winmerge.org/downloads/?lang=en

Quick Note on Naming Files

For our contest system, each problem is given a filename. For example, in the sample contest, one of the problems had a file name of coffee. A solution to this problem must be one of the following files:

```
coffee.py
coffee.java
coffee.cpp
coffee.c
```

In addition, in Java, there is a rule that the only public class in a file must have the same name as the file, so the public class in the submission for coffee.java must be named coffee:

```
public class coffee {
…
}
```

Finally, do NOT include any package statements in your Java file. Often times, if you create a project in Eclipse, a package statement is written in the top of the file. Before submitting, comment this out. Or, if you test via command line compilation, you'll naturally catch this issue.

Printing Floating Point Numbers

Every problem that requires a floating point number output will check accuracy to 10^{-6} absolute or relative precision. Here are default methods of printing out floating point numbers that will work with our judge's tolerance checker:

C++

Please include iomanip and set the precision to at least 7 digits:

```
cout << setprecision(7) << number;
```

Python and Java

Just the default printing via `System.out.println` or `print` will work as this always prints enough digits.

Fast Input/Output

This is only necessary if the input is many megabytes or the output is many megabytes**. In general, it's NOT NECESSARY for easy or medium level problems!!!**

C++ Output

Avoid endl!!! Use '\n' instead because endl flushes the buffer and prints to the screen, and this is slow.

Java Input

There is a class called BufferedReader that reads in a whole line at a time. You can read in the data line by line and then use a StringTokenizer object to parse out each individual token. This is faster than using a Scanner.

Java Output

Instead of writing many prints, store the output in a StringBuilder or StringBuffer object. You can concatenate into the object many times and it does so efficiently. Then, output the final contents of the object with a single System.out.print.

Python Input

This line of code will return a full line of input as a string.

```
sys.stdin.readline().strip()
```

Python Output

This line of code will print out a string s. Note that unlike print, an implicit '\n' character is not added at the end. Also, the input has to be a string so if you want to print a single integer x, you have to convert it to a string using the str function.

```
sys.stdout.write(s)
```

Python Stack Trick

If a recursive program goes deeper than 1000 calls (which isn't much), a python program will crash. To get a deeper call stack do this:

```
import sys
sys.setrecursionlimit(1000000)
```

This simply means that you get a stack depth of 1,000,000.