MANITOBA HIGH SCHOOL PROGRAMMING CONTEST: RESOURCE GUIDE FOR TEACHERS AND STUDENTS (2019)

This guide documents the programming environments that are available to participants in the Manitoba High School Programming Contest held at the University of Manitoba in May 2019. While we make our best effort to ensure that all of the software and versions are available as described below, contest-day changes are sometimes unavoidable.

For each programming language, there is an example of how to read from standard input in the format the contest requires. All contest questions will require one or more values read from standard input. Other forms of input (such as from data files or graphical interfaces) cannot be accepted. Excess output (like prompts for input) may result in a contest submission being rejected.

For input, students should be able to read in integer and decimal values, and strings. In particular, problems may require:

- reading a single numeric or string value on a line of input;
- reading a known number of numeric values per line of input: this may be specified in the problem, or read from input or calculated;
- reading an entire line of input as a string (and performing further processing on it);
- reading a known number of lines of input, either specified in the problem, or read from input or calculated; and
- reading and processing input until a special ("sentinel") value is input; for example, reading positive integers and stopping when a negative value is read. Typically the sentinel value should *not* be processed.

Most environments allow copy and paste of data into standard input; it is a good idea for students to have their test data typed into a text editor so they can avoid re-typing it every time they want to run and test their programs. In addition to the text editing features of the programming environments, the computer labs have the standard Windows Notepad application installed, along with Notepad++. All computers are currently running Windows 7.

All program output must be printed to standard output, as in the examples for each of the supported programming languages given later in this guide. No other forms of output (such as output to files or graphical interfaces) are accepted.

Currently, we support the Java, C/C++, and Python programming languages.

JAVA

The computer systems will have Java 8 installed. It is available from a command prompt for any students who wish to compile and run their programs that way.

There are many ways to read from standard input in Java, but the standard Java Scanner class is typically the easiest. The following example program will read a single integer value from standard input.

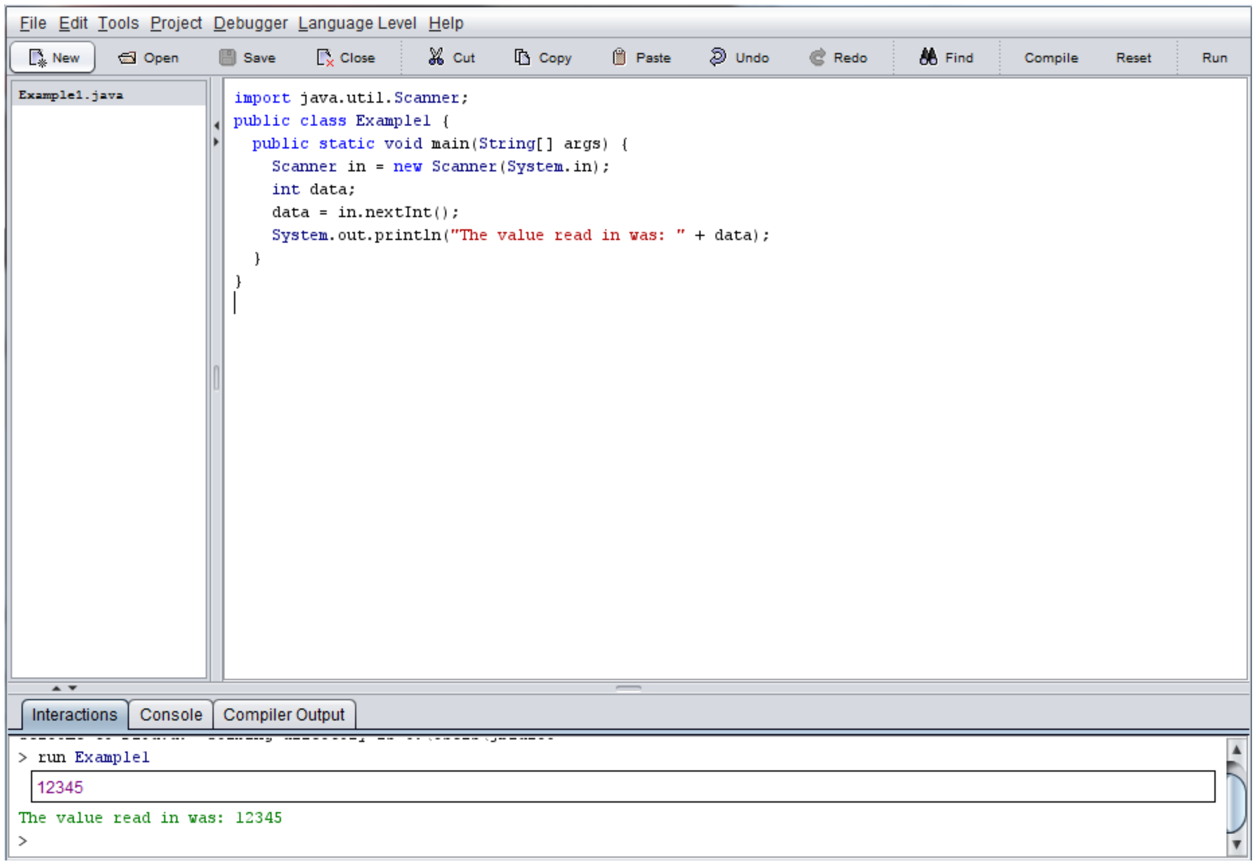
```
import java.util.Scanner;
public class Example1 {
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int data;
    data = in.nextInt();
    System.out.println("The value read in was: " + data);
  }
}
```

Note: to avoid problems, a program should never create more than one Scanner for standard input: don't use `new Scanner(System.in)` more than once in a program. In particular, students should *not* create the Scanner inside a loop; create at the beginning of the program, before any loops begin. Here is an example of reading multiple lines of input, where each line contains an integer value *n* followed by *n* real numbers, exiting when *n* is negative.

```
import java.util.Scanner;
public class Example2 {
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int n = 0
    double data;
    while (n >= 0) {
      n = in.nextInt();
      if (n >= 0) {
        for (int i = 0; i < n; i++) {          // Example input:
          data = in.nextDouble();              //
          System.out.print(data + " ");        // 3 2.5 3.14 -1
        }                                      // 2 1.777778 13.31
        System.out.println();                  // 0
      }                                        // 1 -2718.28182846
    }                                          // 4 3 2 1 0
  }                                            // -1
}
```

DrJava

The most-frequently used Java development environment used in our labs is DrJava. It is the recommended software for our first-year undergraduate students. The labs have the stable 2013 release installed, which only supports Java 7 and 8. No special steps are required in order to execute the contest programs. Students can simply enter and run them.


File Edit Tools Project Debugger Language Level Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile Reset Run
Example1.java
Interactions Console Compiler Output

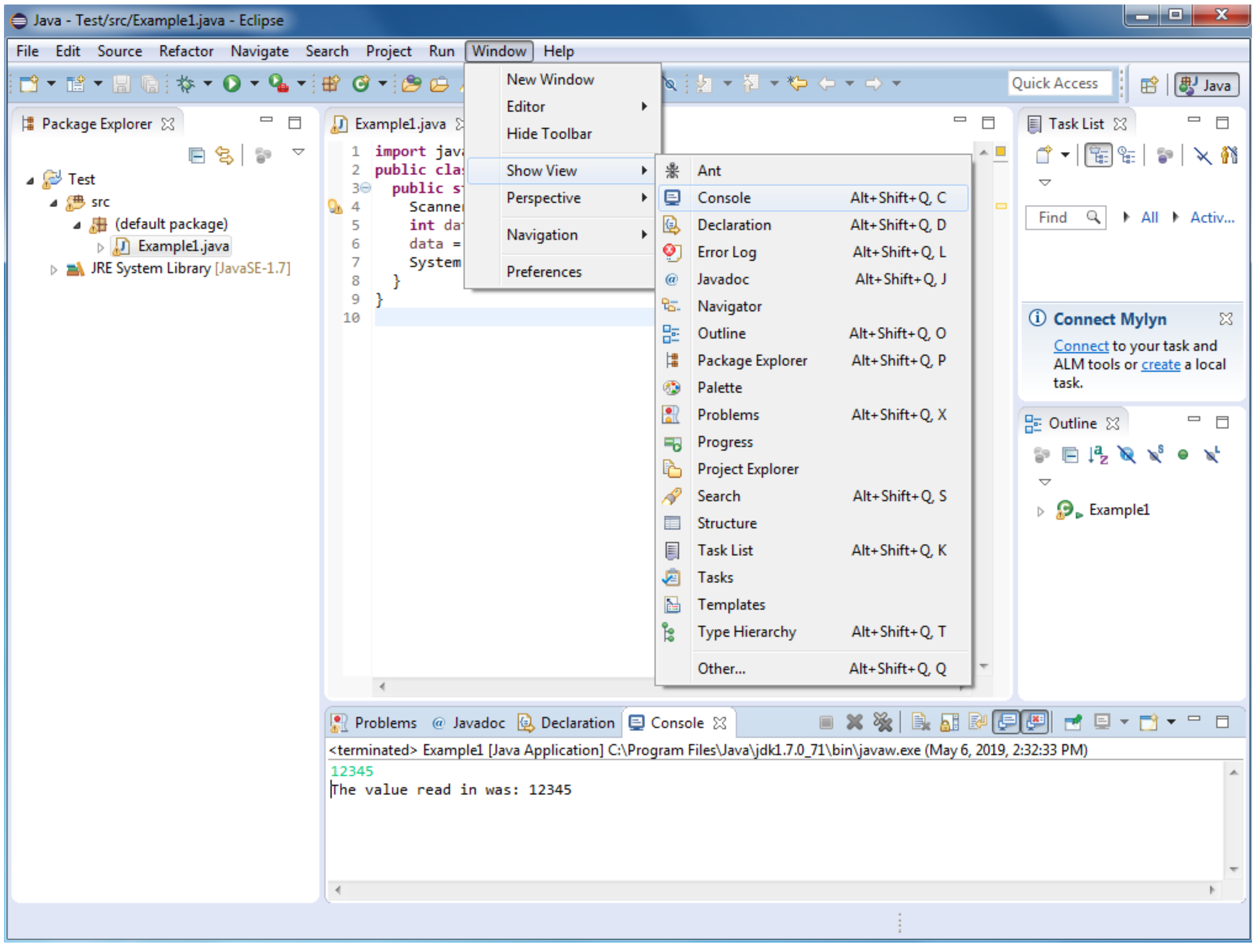
```
import java.util.Scanner;
public class Example1 {
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int data;
    data = in.nextInt();
    System.out.println("The value read in was: " + data);
  }
}
```

```
> run Example1
 12345
The value read in was: 12345
>
```

Eclipse

Eclipse is a more professional development environment (IDE) that provides features to help write code more effectively. Currently, Eclipse Luna (2015) is installed in the labs. Notes for students using Eclipse:

1. When creating a new Java class for a program, keep the “Package” *blank*, even though Eclipse will warn you that “The use of the default package is discouraged”. Packages are *not* allowed for contest submissions.
2. Since contest programs will usually read input before producing any output, Eclipse will not automatically open a window for input. Students will need to open the window before running their program the first time, by selecting Window/Show View/Console.



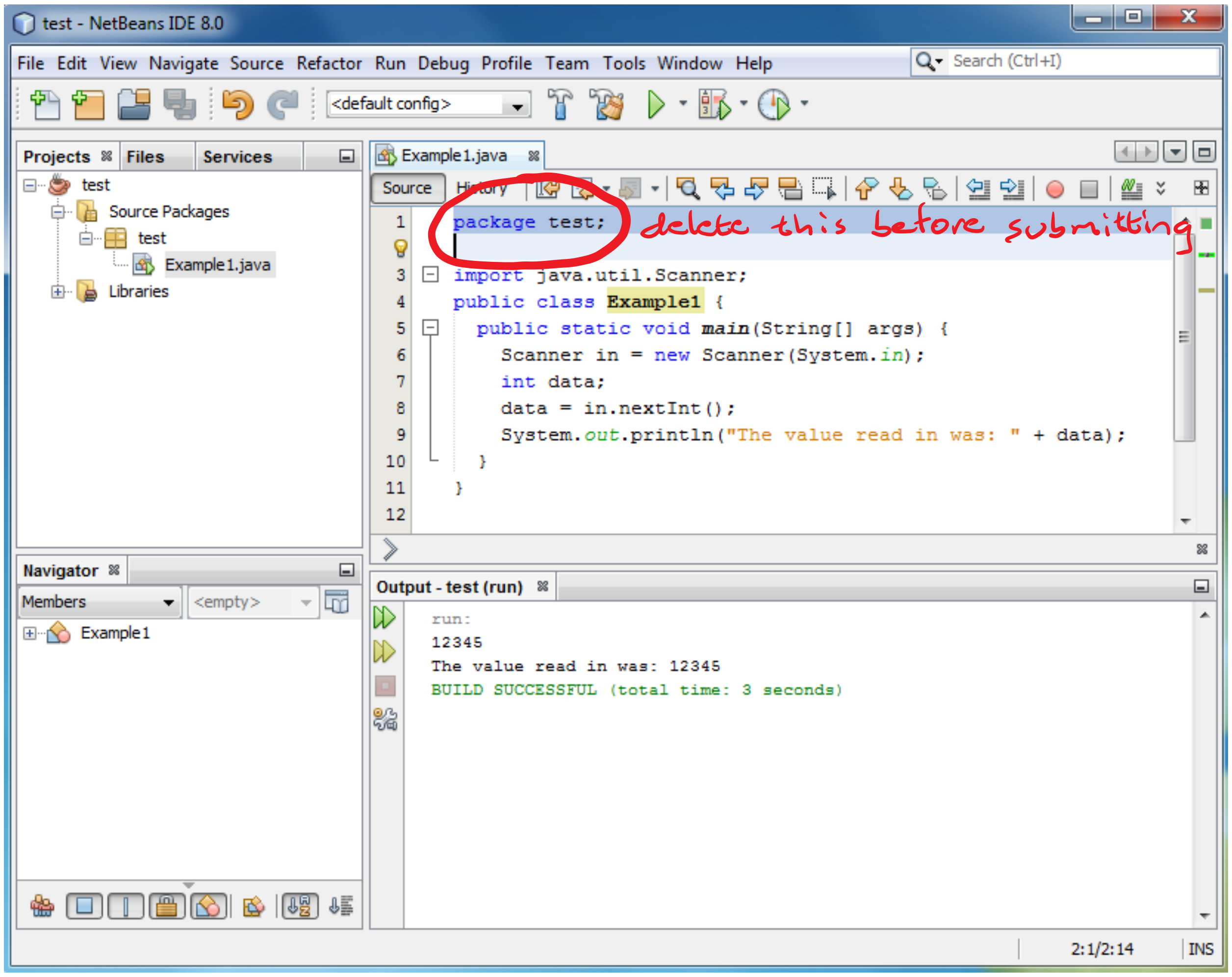
```
import java
public cla
  public s
    Scanne
    int da
    data =
    System
  }
}
```

```
<terminated> Example1 [Java Application] C:\Program Files\Java\jdk1.7.0_71\bin\javaw.exe (May 6, 2019, 2:32:33 PM)
12345
The value read in was: 12345
```

NetBeans

NetBeans is another professional IDE available in our labs. NetBeans version 8 is currently installed. Notes for students using NetBeans:

1. When creating a new Java class, NetBeans will automatically insert a **package** declaration at the top. This can be left in place for running and testing the program. But before submitting their Java source code file using the contest software, students *must delete* this line and save the Java file. They can undo the change if they want to continue to work on the program in NetBeans, and delete it before each submission.



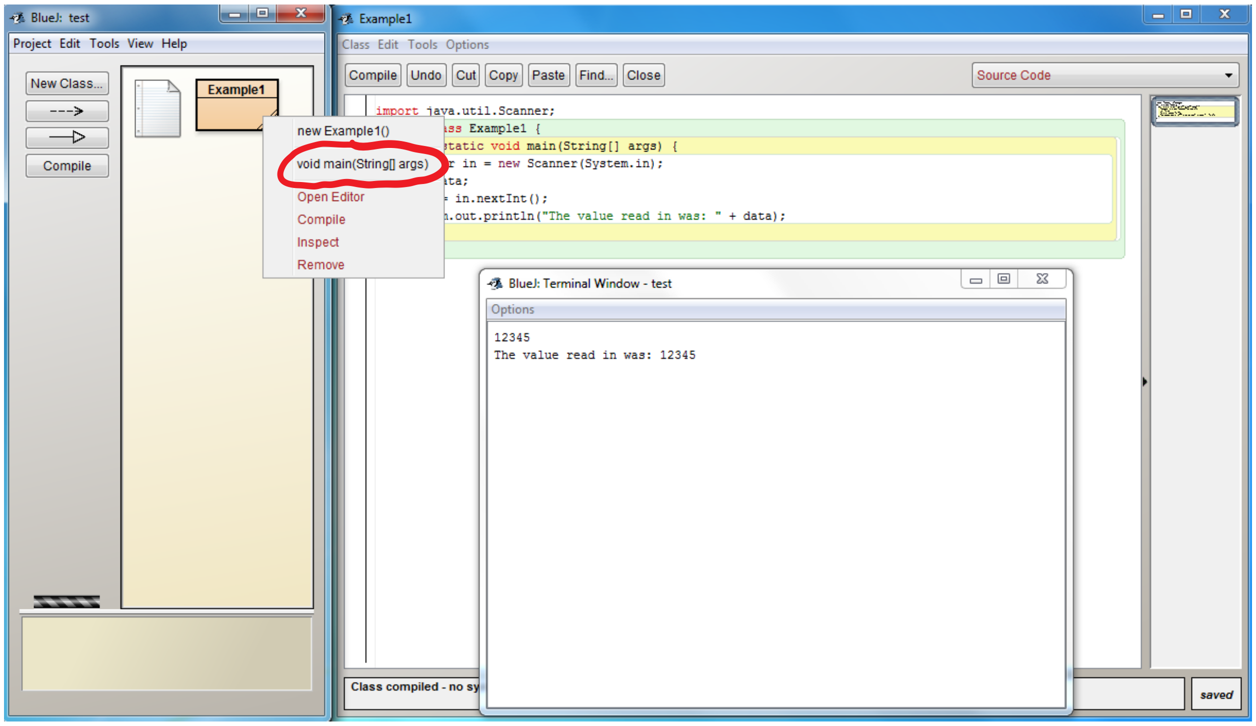
```
package test;

import java.util.Scanner;
public class Example1 {
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int data;
    data = in.nextInt();
    System.out.println("The value read in was: " + data);
  }
}
```

```
run:
12345
The value read in was: 12345
BUILD SUCCESSFUL (total time: 3 seconds)
```

BlueJ

BlueJ is an education-focussed object-oriented Java development environment. Because BlueJ provides a custom interface around "standard" Java, students must do the following in order to create a program that can be accepted by our contest software:

1. The entire solution must be in a single BlueJ class, since the contest software only allows uploading a single Java file.
2. The class must contain a Java main program exactly like the Java example given earlier: `public static void main(String[] args) {` etc. Note that this line must match *exactly*: incorrect names or case will prevent the contest software from running their submission. If students do not typically use main programs, they should bring this reminder with their contest notes.
3. Students should test their program by running the `main` method. It must be the starting point for their solution. Pass no arguments to the `main` method.
4. Input must still come from standard input. Before running the program for the first time, students may need to open the Terminal window (from the View menu).



```
import java.util.Scanner;
          ass Example1 {
          static void main(String[] args) {
          r in = new Scanner(System.in);
          ata;
          = in.nextInt();
          n.out.println("The value read in was: " + data);
```

```
12345
The value read in was: 12345
```

C++

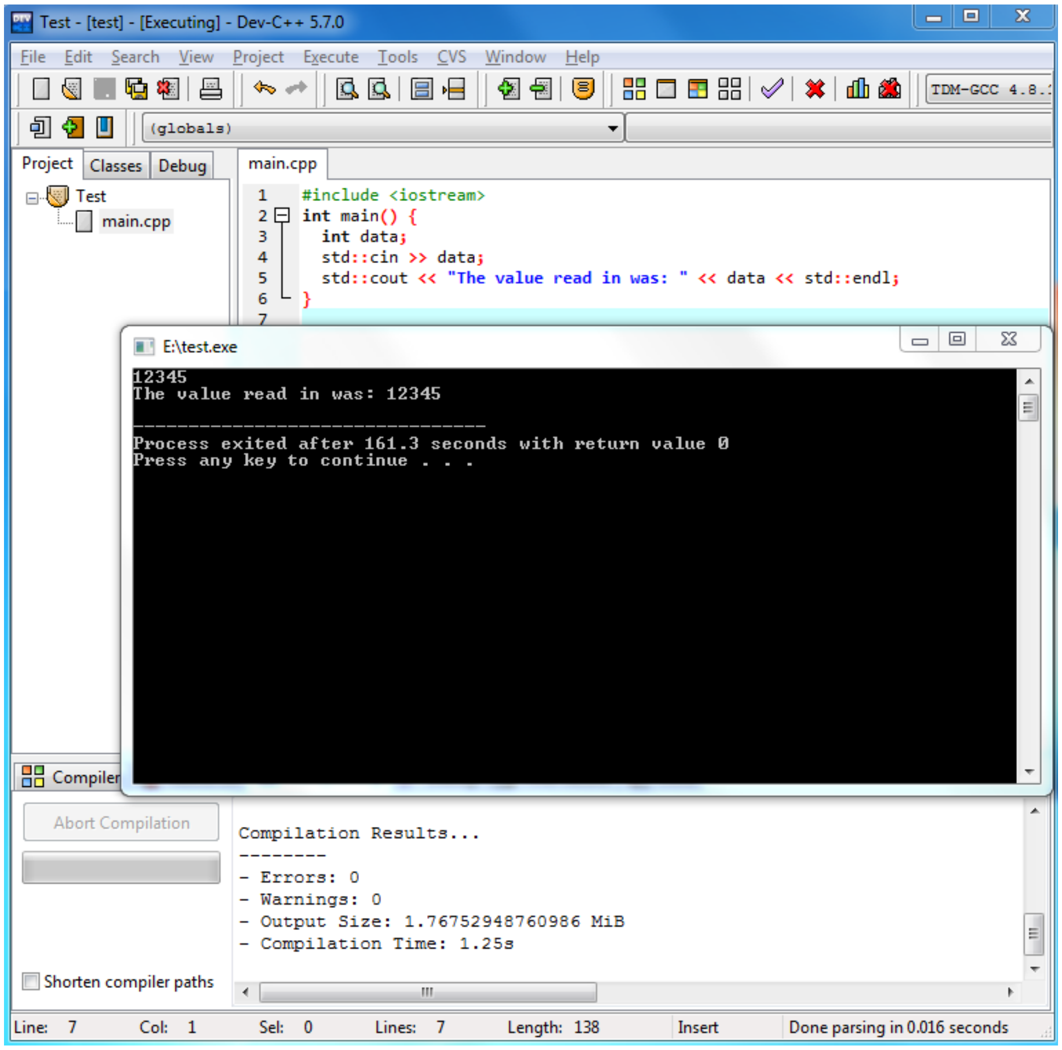
The computer systems have GNU C++ (MinGW) and Microsoft Visual Studio installed.

Students may choose to write programs that use either C or C++ style, but they must be valid C++ programs using the file extension ***filename*.cpp**. Standard input in C++ typically uses **iostream**:

```
#include <iostream>
int main() {
  int data;
  std::cin >> data;
  std::cout << "The value read in was: " << data << std::endl;
}
```

DevC++

Dev-C++ 5.7.0 is installed in the labs. It can build and run C and C++ programs using GNU C and MinGW.



```
#include <iostream>
int main() {
  int data;
  std::cin >> data;
  std::cout << "The value read in was: " << data << std::endl;
}
```

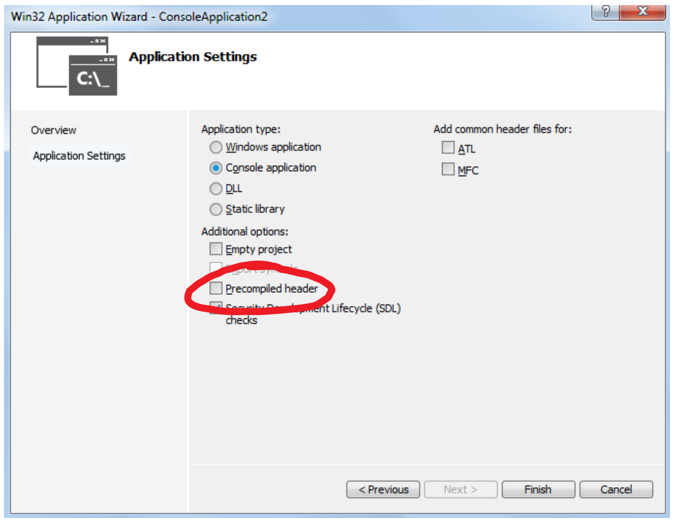
```
12345
The value read in was: 12345

--------------------------------
Process exited after 161.3 seconds with return value 0
Press any key to continue . . .
```

```
Compilation Results...
--------
- Errors: 0
- Warnings: 0
- Output Size: 1.76752948760986 MiB
- Compilation Time: 1.25s
```

Line: 7 Col: 1 Sel: 0 Lines: 7 Length: 138 Insert Done parsing in 0.016 seconds

Microsoft Visual Studio 2012 for C++ is also available. However, submissions must *not* use non-standard Visual Studio C++ extensions are permitted. The following guidelines must be followed:

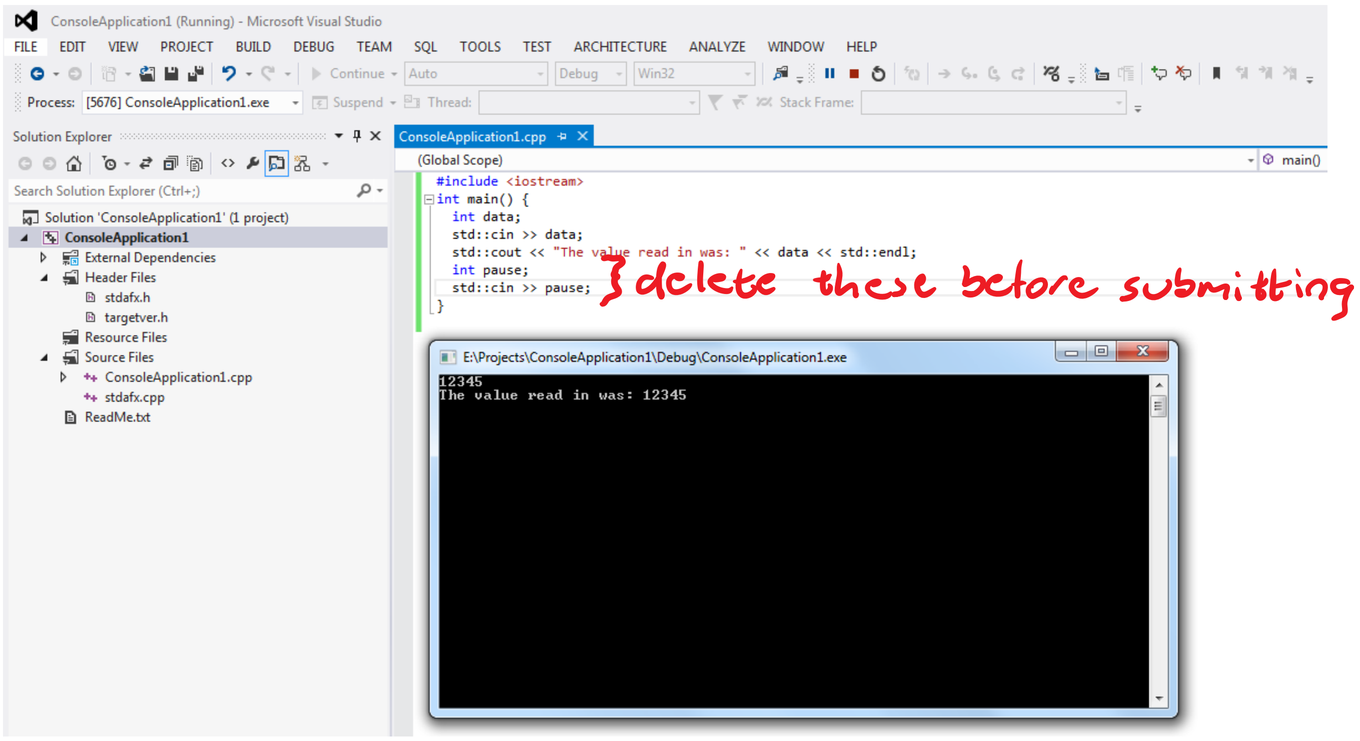
1. There must *not* be an `#include "stdafx.h"` line in the submitted program. This can either be removed before submitting, or students can disable precompiled headers when creating their project, as shown in the image on the right.
2. Use a regular `main` function as the starting point for your program.
3. If you need to add an additional empty input statement at the end of the program to keep the output window open, remove it before submitting your solution.



Application type:
- Windows application
- [✓ Console application]
- DLL
- Static library

Additional options:
- [ ] Empty project
- [ ] Precompiled header
- Security Development Lifecycle (SDL) checks

Add common header files for:
- [ ] ATL
- [ ] MFC


ConsoleApplication1 (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue Auto Debug Win32
Process: [5676] ConsoleApplication1.exe Suspend Thread: Stack Frame:
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ConsoleApplication1' (1 project)
ConsoleApplication1
External Dependencies
Header Files
stdafx.h
targetver.h
Resource Files
Source Files
ConsoleApplication1.cpp
stdafx.cpp
ReadMe.txt
ConsoleApplication1.cpp
(Global Scope)
main()
E:\Projects\ConsoleApplication1\Debug\ConsoleApplication1.exe

```cpp
#include <iostream>
int main() {
   int data;
   std::cin >> data;
   std::cout << "The value read in was: " << data << std::endl;
   int pause;
   std::cin >> pause;
}
```

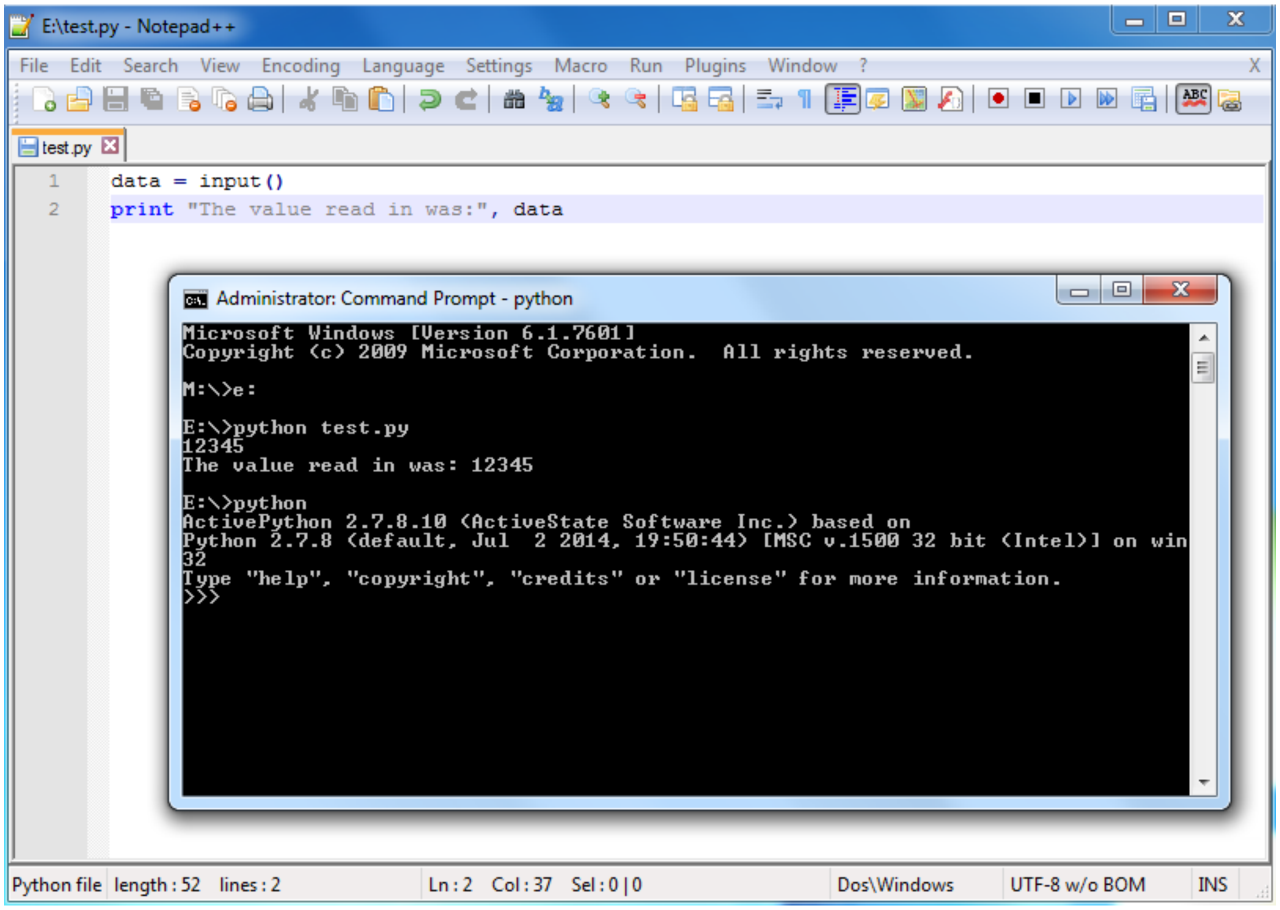
}delete these before submitting

```
12345
The value read in was: 12345
```

No other Visual Studio languages are permitted.

PYTHON

ActivePython 2.7.8.10 is available from the command line. Students will have to use an external editor like Notepad++ to write their programs.


E:\test.py - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
test.py
Administrator: Command Prompt - python

```
data = input()
print "The value read in was:", data
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

M:\>e:

E:\>python test.py
12345
The value read in was: 12345

E:\>python
ActivePython 2.7.8.10 (ActiveState Software Inc.) based on
Python 2.7.8 (default, Jul  2 2014, 19:50:44) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python file length : 52 lines : 2 Ln : 2 Col : 37 Sel : 0 | 0 Dos\Windows UTF-8 w/o BOM INS