Writing standards-compliant C++

We compile and test your programs in Linux (using g++ 3.x) because it’s what we use on our computers. Since many of you use MSVC++ for development, there are often a number of issues in getting your code to compile since VC++ is frequently standards-deficient.

In grading your Assignment 1 submissions, Kris encountered a number of problems while compiling your code. For future assignments, we will offer a 5 point bonus if your code compiles without modification. Here are some tips for writing programs that will work with compilers other than VC++.

- Use \texttt{fabs} for absolute values; on most compilers \texttt{abs} only operates on integers so whatever you pass it will be truncated.

- Variable scoping: you should not do the following:
  \begin{verbatim}
  for(int i = 0; i < something; ++i) { do stuff }
  if(i == x) { do stuff }
  \end{verbatim}

  In other words, a variable declared during \texttt{for} loop initialization goes out of scope after the \texttt{for} loop exits in ISO C++, so this code won’t compile using, e.g., g++.

- Don’t use \texttt{itoa}: such a function doesn’t exist in most libc implementations.

- If you need π, use the following:
  \begin{verbatim}
  #include <math.h>
  #ifndef M_PI
  #define M_PI 3.1415926535897932384626433833
  #endif
  \end{verbatim}

  Most sane \texttt{math.h} implementations define \texttt{M_PI}, but some (MSVC++) do not. Either way, you probably shouldn’t hardcode “3.14” throughout your code since in many cases it’s not nearly accurate enough.

- On any operating system other than windows or with any compiler other than MSVC++, \texttt{#includes} are case sensitive:
  - \texttt{BAD}: \texttt{#include <Math.h>}
  - \texttt{GOOD}: \texttt{#include <math.h>}

- There’s a function called \texttt{random} in most libc implementations, so don’t write your own function called that.

- If you’re sorting \texttt{vector<YourType>}, \texttt{YourType} must have an \texttt{operator<} member, \textit{and} it must be of the form:
  \begin{verbatim}
  bool operator<(const YourType &y) const
  \end{verbatim}

  Many STL implementations require the \texttt{const} declarations.