CS 7A – Spring ’13 Lab 03 Name_____

Each of the following programs has an error. Locate the error, classify it as either a syntax error, a (non-syntax) compile-time error, a link-time error, a run-time error, or a logical error, and then correct it.:

***a.***
```
#include <iostream>
using namespace std;
int main()
{   float x = sqrt(9.11);
    cout << x << endl;
}
```

***b.***
```
#include <iostream>
using namespace std;
int main()
{   int n = 3;
    n /= n*n - 9;
    cout << n << endl;
}
```

***c.***
```
#include <iostream>
using namespace std;
int main()
{   int n *= 4;
    cout << n << endl;
}
```

***d.***
```
#include <iostream>
using namespace std
int main()
{   int n = 314;
    cout << n << endl;
}
```

***e.***
```
#include <iostream>
using namespace std;
int main()
{   float x = 1.10101e28;
    x *= x;
    cout << x << endl;
}
```

***f.***
```
int main()
{   float x = 100.0;
}
```

***g.***
```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{   float x = 200.0;
    cout << pow(x, -x) << endl;
}
```

***h.***
```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{   float x = sqrt(-9.00) ;
    cout << x << endl;
}
```

**Solutions:**

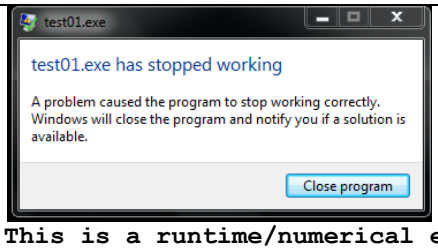
***a.***

```
#include
<iostream>
using namespace
std;
int main()
{   float x = sqrt(9.11);
    cout << x << endl;
}
// sqrt is an undeclared
identifier
```

sqrt is an undeclared identifier and, besides, it's a function, so it requires a definition too. A declaration of sqrt() and a definition for it is found in the cmath library, so if you $include <cmath> the program will compile and run and report the approximate square root of 9.11. As is, the compiler reports,
"error C3861: 'sqrt': identifier not found."

***b.***

```
#include <iostream>
using namespace std;
int main()
{   int n = 3;
    n /= n*n - 9;
    cout << n << endl;
}
// the rhs of n /= nn*n - 9;
// is computed first
(assignment
// operator is right
associative)
// so runtime div by 0 error
```

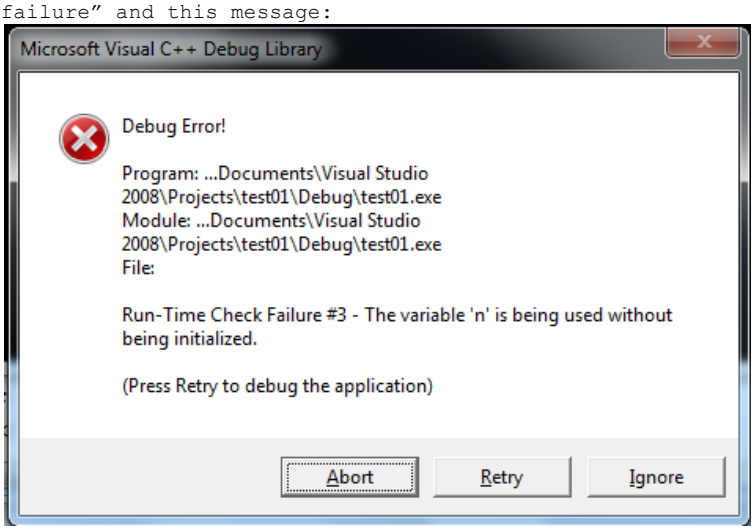



**This is a runtime/numerical error (divide by zero is NAN) that produces the crash message above in MSVC++2008.**

***c.***

```
#include <iostream>
using namespace std;
int main()
{   int n *= 4;
    cout << n << endl;
}  // syntax error since
// since the *= assignment
// operator can't be used
// during initialization
```

MSVC++ produces this error message:
error C2143: syntax error : missing ';' before '*='
Note that if you were to declare, but not initialize n and then multiply it by 4 this way, you'd get a "runtime check failure" and this message:




If you choose "ignore" at this stage it will produce some random number, in the case I checked just now, it was
**858993456**
**Press any key to continue . . .**

***d.***

```
#include <iostream>
using namespace std //;?
int main()
{   int n = 314;
    cout << n << endl;
}
```

**Syntax error: missing semicolon after "using namespace std"**

| | | |
|---|---|---|
| ***e.*** | `#include <iostream>`<br>`using namespace std;`<br>`int main()`<br>`{  float x = 1.10101e28;`<br>`  x *= x;`<br>`  cout << x << endl;`<br>`}`<br>`// The max value for a float`<br>`// with the encoding used in`<br>`// MSVC is`<br>`// 3.402823466e+38`<br>`// x*x > 1.e56 is too big` | **1.#INF**<br>**Press any key to continue . . .**<br>As described in Chapter 1 of the text, a `float` divides its 32 bits into three parts: the leftmost bit is called the ***sign bit,*** the next 8 bits form the ***exponent***, and the right-most 23 bits form the ***fraction***. [..] Note that 127 is subtracted from the stored 8-bit exponent and 1is added to the 23-bit fraction. This algorithm is known as the ***excess-127 floating-point representation.***<br><br>There is a long wiki article on this surprisingly complex issue: **http://en.wikipedia.org/wiki/Floating_point**<br>**This msdn article is more to the point for floats:** **http://msdn.microsoft.com/en-us/library/hd7199ke%28v=vs.80%29.aspx** |
| ***f.*** | `int main()`<br>`{  float x = 100.0;`<br>`}` | `"That is illogical, Captain."  -Spock`<br>`The program is doing nothing except assigning a value to a variable, so "it is useless," according to Hubbard.  I don't know.  It's useful to see that the program compiles and runs.  And it does output to memory, for what that's worth.` |
| ***g.*** | `#include <iostream>`<br>`#include <cmath>`<br>`using namespace std;`<br>`int main()`<br>`{  float x = 200.0;`<br>`  cout << pow(x, -x)`<br>`  << endl;`<br>`}  // use a compiler that`<br>`// implements quadruple`<br>`// precision` | This program compiles ok, but the output is:<br>`x = 0`<br>`Press any key to continue . . .`<br>Exhibiting the run time/numerical error of underflow.  The number `pow(x,-x) = 200^(-200)` is computed by the Windows 7 calculator as<br>`6.22301527786114170714406405378 01e-461`<br>which is significantly less than the minimum computable value of **`1.175494351e-38 for a float.`**<br>Heck, it's even smaller than the minimum computable value for a double: **`2.2250738585072014e-308`** |
| ***h.*** | `include <iostream>`<br>`#include <cmath>`<br>`using namespace std;`<br>`int main()`<br>`{  float x = sqrt(-9.00) ;`<br>`  cout << x << endl;`<br>`}` | `Msvc produces this error message:`<br><br>`error C2668: 'sqrt' : ambiguous call to overloaded function`<br>`could be 'long double sqrt(long double)'`<br>`or  'float sqrt(float)'`<br>`or  'double sqrt(double)'`<br><br>`Evidently, the output of sqrt() can *not* be complex or imaginary, but the error message doesn't convey that very well.` http://stackoverflow.com/questions/10145295/removing-ambiguity-when-using-sqrt-function-in-a-template-class |