Homework

15

A Generic to_string

Using the `to_string()` examples from the reading, write a generic version, and place it into the `h15.h` header file.

- Make sure you surround it with the `#if` preprocessor directive, so it will only be included when compiling with early versions of C++.
- **Fully qualify** each library name.
- Do not put `using namespace` in your header file.
- Add your id to `h15.cpp`. This will be the only code inside `h15.cpp`.

Do `make test` to check your code. This will compile your code using C++98 and C++17, and then test the functions using C++11. Unfortunately, all does not seem perfect.

```
Testing using C++98
----------------------------------------
Checking: to_string function ------------------
   + to_string(42)->"42"
   X to_string(3.F): expected [3.000000] but found [3]
   + to_string(-1U)->"4294967295"
   + to_string(4L)->"4"
   X to_string('c'): expected [99] but found [c]
   X to_string(1.7L): expected [1.700000] but found [1.7]
   X to_string(2.7e3): expected [2700.000000] but found [2700]
   + to_string(2.17e-4)->"0.000217"
   + to_string(1.0/0.0)->"inf"
   + to_string(-1.0/0.0)->"-inf"
   + to_string(0.0/0.0)->"nan"
------------------------------------------------------------
  Tests passing 7/11 (64%).
```

We want our code to act exactly like the new, standard library version of this functions. The expected values above are what the **standard library** returns when run with each of these tests. (We're using the C++17 library as a test oracle here.)

The three floating-point numbers errors are easy to understand. It looks like floating-point numbers are converted using **`fixed`** notation, even if they were originally otherwise. Simply change one line to solve this problem.

```
out << std::fixed << value;
```

Now, you should have about 91%. The only line that fails is the **`char`** parameter. It seems that the standard library version wants to print the **`ASCII`** value, while our template version wants to print the actual character.

How to we fix this? The easiest way is to print the expression **`(0 + value)`** instead of just **`value`**. When **`0`** is added to a **`char`**, the result is an **`int`**, and we get the same result as the standard library.

Now you can do **`make test`**, and, if you get 100%, then **`make submit`**. Visit me in my office or on Piazza if you need help.