CS 372 Software Construction

In-Class Activity: Code Review

Perform a code review of file `lowpos.h` and the unit tests for function `lowPositive`. Code is on the following pages, and also linked under the course materials tab on Blackboard.

Review Criteria

- Does the code conform to standards (style guidelines, etc.)?
- Does the code contain any logic errors?
- Is the code *sufficient*? Does it allow the project to fully meet the requirements that motivated the change?
- Are the associated tests sufficient?
  - Do new tests need to be written, to handle the changed code?
  - Do existing tests need to be rewritten, to account for the changes?

Project Coding Standards

- All code must be in C++, following the 2011 ANSI/ISO Standard.
- Standard conventions must be followed regarding use of header and source files, `#ifndef`, etc.
- All identifiers must consist of lower-case letters and digits, with words separated by underscores.
- Each block must be enclosed by braces.
- Each indentation level will be three blanks. Tabs will not be used.
- Each function and class definition is preceded by explanatory comments.
- Function comments must include preconditions and postconditions.
- Comments on each class template or function template must include requirements on types, for template parameter types.
- Each non-parameter variable or data member must be commented at its declaration, indicating the meaning of the variable's value.

Relevant Portion of Design Document

2.3.1. Function template `lowPositive`.

- Prototype:
- `template<typename FDIter>`
- `int lowPositive(FDIter first, FDiter last);`
- Declared and defined in header `lowpos.h`.
- Takes a range specified by two forward iterators. Returns the least positive value in the range, or `0` if the range contains no positive values.

Author Notes for Reviewer

- Wrote header `lowpos.h`, including function template `lowPositive`.
- Relevant unit tests added to test code. Existing unit tests unchanged.
- Done in accordance with design spec 2.3.1.
- Note: I have not done much with iterators before. It passes all the tests, but please check that I have handled the iterators correctly.

File `lowpos.h`

```
// lowpos.h
// Glenn G. Chappell
// 29 Feb 2016
//
// Header for function lowPositive


// lowPositive
// Given range of int values, return least positive value in the range,
// or 0 if the range contains no positive value.
// Preconditions:
//     [first, last) is a valid range.
// Postconditions:
//     The return value is the least positive value in the range, or 0
//      if the range contains no positive value.
// Requirements on Types:
//     FDIter is a forward iterator type.
//     The value type of FDIter is int.
// Since this is a template, we define it here, and not in a .cpp file.
template <typename FDIter>
int lowPositive(FDIter first, FDIter last)
{
    int minval;  // Holds least positive value so far, or zero if none

    // Initialize minval appropriately
    if (*first > 0)
        minval = *first;
    else
        minval = 0;

    // Go through each item in range, updating minval as needed
    for (FDIter it = first;
         it != last;
         ++it)
    {
        if (*it <= 0)  // Zero or negative? Skip it
            continue;
        if (*it < minval)  // Positive & below minval? Update minval
            minval = *it;
    }

    // Done; minval is our result
    return minval;
}
```

Relevant Portion of Unit-Testing Code (Uses *Catch* framework)

```
TEST_CASE("Single values: positive", "[lowpos]")
{
    vector<int> v1 = { 42 };
    REQUIRE(lowPositive(v1.begin(), v1.end()) == 42);

    vector<int> v2 = { 1 };
    REQUIRE(lowPositive(v2.begin(), v2.end()) == 1);
}


TEST_CASE("Single values: zero or negative", "[lowpos]")
{
    vector<int> v1 = { 0 };
    REQUIRE(lowPositive(v1.begin(), v1.end()) == 0);

    vector<int> v2 = { -42 };
    REQUIRE(lowPositive(v2.begin(), v2.end()) == 0);
}


TEST_CASE("Small ranges, containing positive", "[lowpos]")
{
    vector<int> v1 = { 15, -3, 20, 0, -2, 8, 14 };
    REQUIRE(lowPositive(v1.begin(), v1.end()) == 8);

    vector<int> v2 = { 10, 11, 12, 11, 10, 9, 10 };
    REQUIRE(lowPositive(v2.begin(), v2.end()) == 9);
}


TEST_CASE("Small ranges, not containing positive", "[lowpos]")
{
    vector<int> v1 = { -10, -20, -30, -5, -100 };
    REQUIRE(lowPositive(v1.begin(), v1.end()) == 0);

    vector<int> v2 = { -1, -2, 0, -100, -3 };
    REQUIRE(lowPositive(v2.begin(), v2.end()) == 0);

    vector<int> v3 = { 0, 0, 0, 0, 0 };
    REQUIRE(lowPositive(v2.begin(), v2.end()) == 0);
}
```

(cont'd next page)

Relevant Portion of Unit-Testing Code (cont'd)

```
TEST_CASE("Large range, containing positive", "[lowpos]")
{
    vector<int> v1;
    const int bigval = 50000;  // For making other values
    for (int i = 0; i < bigval; ++i)
    {
        v1.push_back(4*bigval-2*i);
        v1.push_back(-2*bigval-2*i);
        v1.push_back(3*bigval+2*i);
        v1.push_back(-4*bigval+2*i);
    }
    REQUIRE(lowPositive(v1.begin(), v1.end()) == 2*bigval+2);
}


TEST_CASE("Large range, not containing positive", "[lowpos]")
{
    vector<int> v1;
    const int bigval = 50000;  // For making other values
    for (int i = 0; i < bigval; ++i)
    {
        v1.push_back(-2*bigval-2*i);
        v1.push_back(-4*bigval+2*i);
    }
    REQUIRE(lowPositive(v1.begin(), v1.end()) == 0);
}


TEST_CASE("Other kinds of ranges", "[lowpos]")
{
    // std::deque using reverse iterators, one item skipped
    deque<int> dd = { 10, -9, 5, 0, 20, 1 };
    REQUIRE(lowPositive(dd.rbegin()+1, dd.rend()) == 5);

    // std::forward_list
    forward_list<int> fl = { 10, -9, 5, 0, 20 };
    REQUIRE(lowPositive(fl.begin(), fl.end()) == 5);
}
```