CS 200

Libby Wichman

InClassLab8: Testing

# Average

Test Values:

// Test 1 = 1, 1, 1

Output should be 1

// Test 2 = -1, 0, 1

Output should be 0

// Test3 = 3, 6, 9

Output should be 6

void Test\_Average()

{

// Test 1

int t1Input1 = 1, t1Input2 = 1, t1Input3 = 1;

int t1ExpectedOutput = 1;

int t1ActualOutput = Average(t1Input1, t1Input2, t1Input3);

if (t1ExpectedOutput != t1ActualOutput)

{

cout << "Failed for " << t1Input1 << ", " << t1Input2 << ", " << t1Input3 << endl;

}

else

{

cout << "Passed for " << t1Input1 << ", " << t1Input2 << ", " << t1Input3 << endl;

}

// Test 2

int t2Input1 = -1, t2Input2 = 0, t2Input3 = 1;

int t2ExpectedOutput = 0;

int t2ActualOutput = Average(t2Input1, t2Input2, t2Input3);

if (t2ExpectedOutput != t2ActualOutput)

{

cout << "Failed for " << t2Input1 << ", " << t2Input2 << ", " << t2Input3 << endl;

}

else

{

cout << "Passed for " << t2Input1 << ", " << t2Input2 << ", " << t2Input3 << endl;

}

// Test 3

int t3Input1 = 3, t3Input2 = 6, t3Input3 = 9;

int t3ExpectedOutput = 6;

int t3ActualOutput = Average(t3Input1, t3Input2, t3Input3);

if (t3ExpectedOutput != t3ActualOutput)

{

cout << "Failed for " << t3Input1 << ", " << t3Input2 << ", " << t3Input3 << endl;

}

else

{

cout << "Passed for " << t3Input1 << ", " << t3Input2 << ", " << t3Input3 << endl;

}

}

Within the Average function, the return needed to have parenthesis around “a + b + c”. This ensures that the order of operations will sum the values prior to dividing by the total number of values summed.

# Max

Test Values:

// Test 1 = 1, 1, 1

Output should be 1

// Test 2 = -5, 0, 5

Output should be 5

// Test3 = 30, 20, 10

Output should be 30

void Test\_Max()

{

// Test 1

int t1Input1 = 1, t1Input2 = 1, t1Input3 = 1;

int t1ExpectedOutput = 1;

int t1ActualOutput = Max(t1Input1, t1Input2, t1Input3);

if (t1ExpectedOutput != t1ActualOutput)

{

cout << "Failed for " << t1Input1 << ", " << t1Input2 << ", " << t1Input3 << endl;

}

else

{

cout << "Passed for " << t1Input1 << ", " << t1Input2 << ", " << t1Input3 << endl;

}

// Test 2

int t2Input1 = -5, t2Input2 = 0, t2Input3 = 5;

int t2ExpectedOutput = 5;

int t2ActualOutput = Max(t2Input1, t2Input2, t2Input3);

if (t2ExpectedOutput != t2ActualOutput)

{

cout << "Failed for " << t2Input1 << ", " << t2Input2 << ", " << t2Input3 << endl;

}

else

{

cout << "Passed for " << t2Input1 << ", " << t2Input2 << ", " << t2Input3 << endl;

}

// Test 3

int t3Input1 = 30, t3Input2 = 20, t3Input3 = 10;

int t3ExpectedOutput = 30;

int t3ActualOutput = Max(t3Input1, t3Input2, t3Input3);

if (t3ExpectedOutput != t3ActualOutput)

{

cout << "Failed for " << t3Input1 << ", " << t3Input2 << ", " << t3Input3 << endl;

}

else

{

cout << "Passed for " << t3Input1 << ", " << t3Input2 << ", " << t3Input3 << endl;

}

}

Within the Max function, I added a bit more logic to ensure it would be able to output the Max value of the arguments, because the original function only tested whether a was less than b. In the first part of the if statement, the original function only tested if a was less than b, but it did not test if the third argument was less than b as well. In order to ensure the return value of the first block was b, I added && (c < b). In the second block, I added the test condition a < c to ensure c was the return value. For the last block, I added an else with a return a.

# Factorial

Test Values:

// Test 1 = 0

Output should be 1

// Test 2 = 5

Output should be 120

// Test3 = 4

Output should be 24

void Test\_Factorial()

{

// Test 1

int t1Input1 = 0;

int t1ExpectedOutput = 1;

int t1ActualOutput = Factorial(t1Input1);

if (t1ExpectedOutput != t1ActualOutput)

{

cout << "Failed for " << t1Input1 << endl;

}

else

{

cout << "Passed for " << t1Input1 << endl;

}

// Test 2

int t2Input1 = 5;

int t2ExpectedOutput = 120;

int t2ActualOutput = Factorial(t2Input1);

if (t2ExpectedOutput != t2ActualOutput)

{

cout << "Failed for " << t2Input1 << endl;

}

else

{

cout << "Passed for " << t2Input1 << endl;

}

// Test 3

int t3Input1 = 4;

int t3ExpectedOutput = 24;

int t3ActualOutput = Factorial(t3Input1);

if (t3ExpectedOutput != t3ActualOutput)

{

cout << "Failed for " << t3Input1 << endl;

}

else

{

cout << "Passed for " << t3Input1 << endl;

}

}

Instead of setting the counter to n, I set it to n-1. This ensures that the starting value is not multiplied twice during the loop.

# FormatName

Test Values:

// Test 1 = Albus, Dumbledore

Output should be “Dumbledore, Albus”

// Test 2 = Severus, Snape

Output should be “Snape, Severus”

// Test3 = Harry, Potter

Output should be “Potter, Harry”

void Test\_FormatName()

{

// Test 1

string t1Input1 = "Albus", t1Input2 = "Dumbledore";

string t1ExpectedOutput = "Dumbledore, Albus";

string t1ActualOutput = FormatName(t1Input1, t1Input2);

if (t1ExpectedOutput != t1ActualOutput)

{

cout << "Failed for " << t1Input1 << ", " << t1Input2 << endl;

}

else

{

cout << "Passed for " << t1Input1 << ", " << t1Input2 << endl;

}

// Test 2

string t2Input1 = "Severus", t2Input2 = "Snape";

string t2ExpectedOutput = "Snape, Severus";

string t2ActualOutput = FormatName(t2Input1, t2Input2);

if (t2ExpectedOutput != t2ActualOutput)

{

cout << "Failed for " << t2Input1 << ", " << t2Input2 << endl;

}

else

{

cout << "Passed for " << t2Input1 << ", " << t2Input2 << endl;

}

// Test 3

string t3Input1 = "Harry", t3Input2 = "Potter";

string t3ExpectedOutput = "Potter, Harry";

string t3ActualOutput = FormatName(t3Input1, t3Input2);

if (t3ExpectedOutput != t3ActualOutput)

{

cout << "Failed for " << t3Input1 << ", " << t3Input2 << endl;

}

else

{

cout << "Passed for " << t3Input1 << ", " << t3Input2 << endl;

}

}

int main()

{

Test\_Average();

Test\_Max();

Test\_Factorial();

Test\_FormatName();

while (true)

{

; // keep it open

}

return 0;

}

To correct the function, I added a space to the string output between the last and first variable in the return statement.