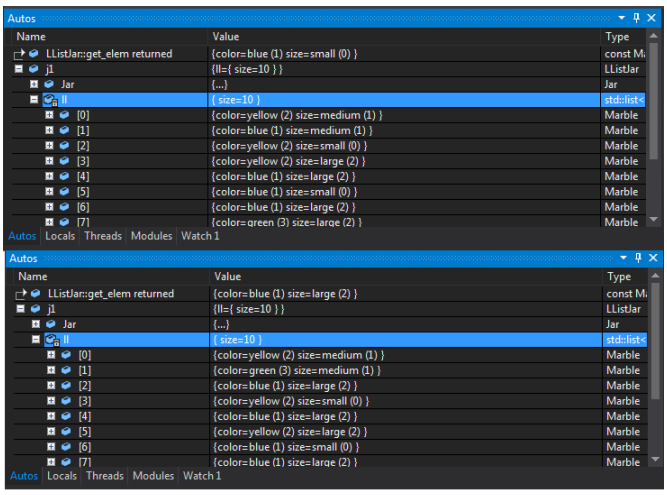
LList_Jar Debugging
Due Feb 5 at midnight to CSNet

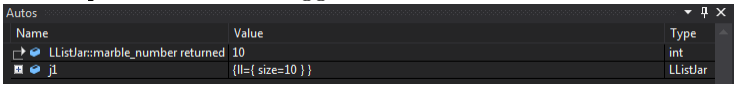
Arnaldo Villarreal ,UIN: 820006691

Avreal31 E-mail: villarreal_alan@neo.tamu.edu

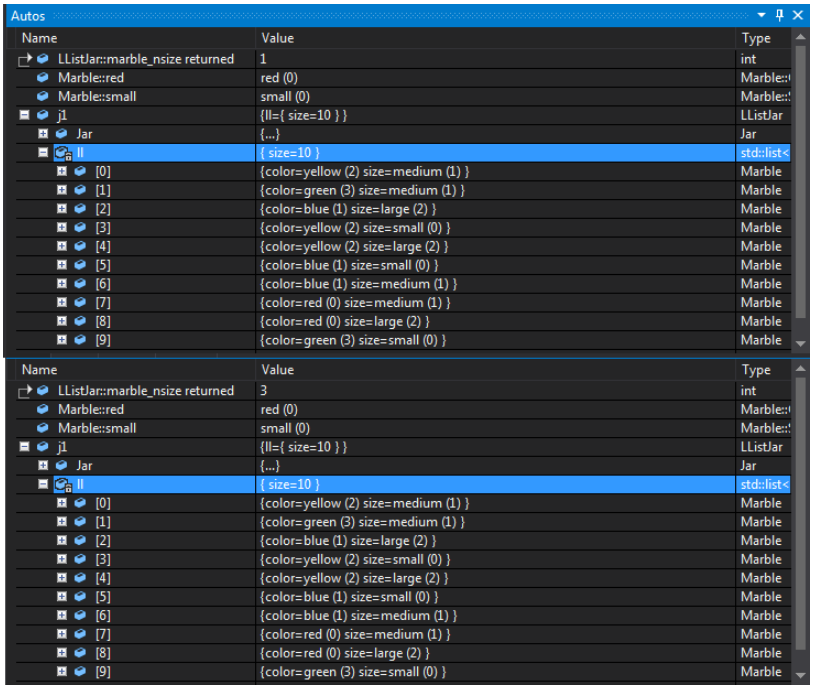
For this assignment I used the debugger built in to Microsoft Visual Studio 2012 to find the logic errors within the LList_jar code. First I cleared the code of any compilation errors, the only one being an undefined exception class. Then the debugging process began. I made a test file called LLJar_main.cpp in which I initialized 3 jars and a marble to conduct tests with. The default constructor was not defined, but thats not a bug but rather just something left out (which I defined). The regular constructer worked fine. The copy constructor however, caused a program crash. By using the 'step into' function of the debugger, I went line by line into each subroutine in the copy constructor, and found that in the function get_elem(), line 39 (specifically the part ll.begin()–) casued this crash. I removed the – thinking that it would solve the issue, but instead of showing the i*th* element, it showed the i*th*+1, for example get_elem(4) got the the 5th element. So once again I went into the get_elem() function and found that *advance(it,i)* casued this, so I changed it to *advance(it,i-1)*. It is a scrappy solution, but operators on the ll would not work as I wanted them to. The following image shows the reult of changing get_elem(), this is the debugging info window in Visual Studio. The top part is the original get_elem() function, and the second is after my fix.



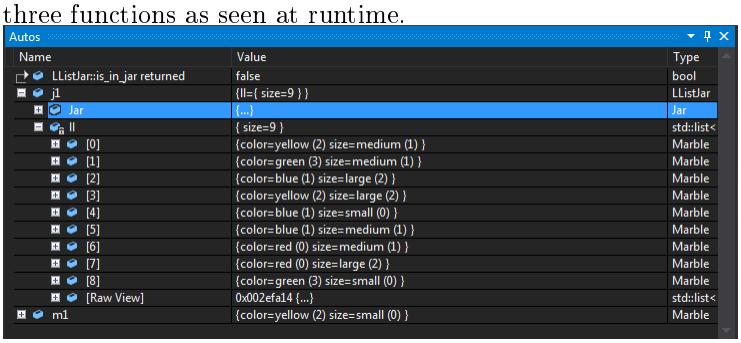
Fixing the get_elem() function was very important, as it is used in most of the other functions. Unfortunately, as a result of my fix, all uses of get_elem(i) and get_elem(j) had to be changed to get_elem(i+1) and get_elem(j+1) respectively. Once that was settled I was able to start testing the other functions. I created a jar with 10 marbles, and ran the marble_number() function, which had no runtime errors, here is the output from the debugger.



Following that I tested the Marble_nsize() and Marble_ncolor() functions, which return the number of marbles of a specific size/color. I noticed while running line-by-line in the debugger that because the pop_front() function was being used, the size of the list was being changed while still looping, so before the loop i created int realsize = temp.size(). Using this allowed the loop to go through the proper amount of iterations. The following image shows the before after of the nColor function when checking for the number of red marbles.

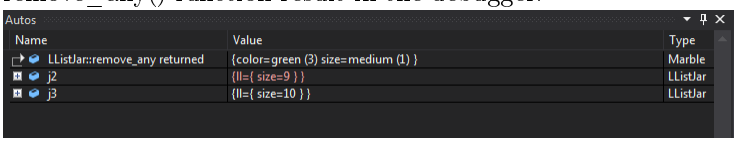


The next function that needed fixing was is_in_jar(). The debugger reported a crash once again because of ll.begin()–, it was unecessary so I removed it. After fixing is_in_jar() I went to add() and remove(), these functions check if the marble is in the jar so it was important that is_in_jar was implemented correctly. I didn't need the debugger for add(), as it was very obvious from looking at it that it was missing a '!' in the if statement. The remove() function required no changes. The following image actually demonstrates all three functions as seen at runtime.



In this image, the function add(m1) was used, but the marble already existed so nothing happened, then it was removed, and finally is_in_jar(m1) returned false, as I had just removed m1 from the jar. The clear() function required no changes, as it worked as intended right away. The same goes for is_empty().

For the next set of tests, I created a new LListJar with 10 elements, I then called the LListJar constructor with a new jar, j3, as the argument to test the copy constructor. There was no faulty logic there either. The remove_any() function was quite troubling for me, as I had to make use of breakpoints with in the function to find the exact spot where it failed as I kept getting range errors in my list. In particular the line "advance(it, rand() % ll.size());" caused a crash, to remedy this i created an int intitialized to "rand() % (ll.size()-1)", which is still a random number, and it worked with my list. The following image shows the remove_any() function result in the debugger.



The final 3 functions were the set operations, since merge depended on difference I started with difference. First I initialized 2 jars, j1 and j2, with 10 marbles each, then created an empty jar j3. Because of the nested structure, it was dificult to debug. After going line by line and 'stepping into' the loop bodies, I found that the break statement was causing issues, so i removed it. Also the boolean value of found was resulting in nothing being returned, so I moved that to the outside of the 2nd if statement. The following image shows

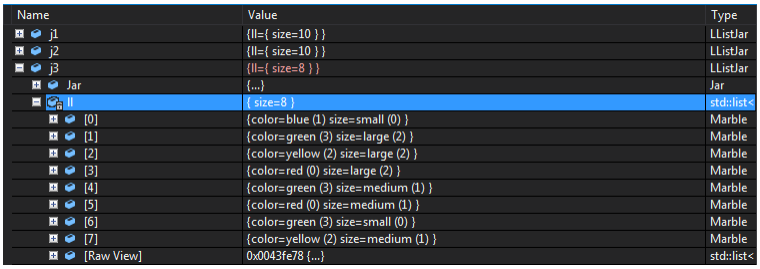
the result.

Name	Value	Type
j1	{ll={ size=10 } }	LListJar
Jar	{...}	Jar
ll	{ size=10 }	std::list<
[0]	{color=blue (1) size=small (0) }	Marble
[1]	{color=green (3) size=large (2) }	Marble
[2]	{color=yellow (2) size=large (2) }	Marble
[3]	{color=red (0) size=large (2) }	Marble
[4]	{color=blue (1) size=large (2) }	Marble
[5]	{color=green (3) size=medium (1) }	Marble
[6]	{color=yellow (2) size=small (0) }	Marble
[7]	{color=red (0) size=medium (1) }	Marble
[8]	{color=green (3) size=small (0) }	Marble
[9]	{color=yellow (2) size=medium (1) }	Marble
[Raw View]	0x0020f8b4 {...}	std::list<
j2	{ll={ size=10 } }	LListJar
Jar	{...}	Jar
ll	{ size=10 }	std::list<
[0]	{color=yellow (2) size=medium (1) }	Marble
[1]	{color=green (3) size=large (2) }	Marble
[2]	{color=green (3) size=medium (1) }	Marble
[3]	{color=red (0) size=large (2) }	Marble
[4]	{color=red (0) size=medium (1) }	Marble
[5]	{color=blue (1) size=medium (1) }	Marble
[6]	{color=green (3) size=small (0) }	Marble
[7]	{color=red (0) size=small (0) }	Marble
[8]	{color=blue (1) size=small (0) }	Marble
[9]	{color=yellow (2) size=large (2) }	Marble
[Raw View]	0x0020f89c {...}	std::list<
j3	{ll={ size=2 } }	LListJar
Jar	{...}	Jar
ll	{ size=2 }	std::list<
[0]	{color=blue (1) size=large (2) }	Marble
[1]	{color=yellow (2) size=small (0) }	Marble
[Raw View]	0x0020f884 {...}	std::list<

Following this, I completed the merge function, the only thing required in this was changing get_elem(i) to get_elem(i+1). The following image shows the result of the merge.

Name	Value
j1	{ll={ size=10 } }
Jar	{...}
__vfptr	0x013ffdf4 {JarDebug.exe!const LListJa
ll	{ size=10 }
[0]	{color=blue (1) size=small (0) }
[1]	{color=green (3) size=large (2) }
[2]	{color=yellow (2) size=large (2) }
[3]	{color=red (0) size=large (2) }
[4]	{color=blue (1) size=large (2) }
[5]	{color=green (3) size=medium (1) }
[6]	{color=yellow (2) size=small (0) }
[7]	{color=red (0) size=medium (1) }
[8]	{color=green (3) size=small (0) }
[9]	{color=yellow (2) size=medium (1) }
[Raw View]	0x0014fd28 {...}
j2	{ll={ size=10 } }
Jar	{...}
ll	{ size=10 }
[0]	{color=yellow (2) size=medium (1) }
[1]	{color=green (3) size=large (2) }
[2]	{color=green (3) size=medium (1) }
[3]	{color=red (0) size=large (2) }
[4]	{color=red (0) size=medium (1) }
[5]	{color=blue (1) size=medium (1) }
[6]	{color=green (3) size=small (0) }
[7]	{color=red (0) size=small (0) }
[8]	{color=blue (1) size=small (0) }
[9]	{color=yellow (2) size=large (2) }
[Raw View]	0x0014fd10 {...}
j3	{ll={ size=12 } }
Jar	{...}
ll	{ size=12 }
[0]	{color=blue (1) size=large (2) }
[1]	{color=yellow (2) size=small (0) }
[2]	{color=yellow (2) size=medium (1) }
[3]	{color=green (3) size=large (2) }
[4]	{color=green (3) size=medium (1) }
[5]	{color=red (0) size=large (2) }
[6]	{color=red (0) size=medium (1) }
[7]	{color=blue (1) size=medium (1) }
[8]	{color=green (3) size=small (0) }
[9]	{color=red (0) size=small (0) }
[10]	{color=blue (1) size=small (0) }
[11]	{color=yellow (2) size=large (2) }
[Raw View]	0x0014fcf8 {...}

The final function that needed to be done was the intersect, by manually iterating through the loop I found that the break statement caused issues so i removed it, also 'add(jar1.get_elem(i));' needed to be changed to 'add(jar2.get_elem(i))'. Here is the reult of the intersection.



Name	Value	Type
j1	{ll={ size=10 } }	LListJar
j2	{ll={ size=10 } }	LListJar
j3	{ll={ size=8 } }	LListJar
Jar	{...}	Jar
ll	{ size=8 }	std::list<
[0]	{color=blue (1) size=small (0) }	Marble
[1]	{color=green (3) size=large (2) }	Marble
[2]	{color=yellow (2) size=large (2) }	Marble
[3]	{color=red (0) size=large (2) }	Marble
[4]	{color=green (3) size=medium (1) }	Marble
[5]	{color=red (0) size=medium (1) }	Marble
[6]	{color=green (3) size=small (0) }	Marble
[7]	{color=yellow (2) size=medium (1) }	Marble
[Raw View]	0x0043fe78 {...}	std::list<

The j1 and j2 arguments are the same as in the other set.