This is a graded discussion: **5 points possible**

due Sep 11

36 99

Assignment 3 Discussion

Post your contribution to the assignment 3 discussion here. This could involve asking a question, answering another student's question, giving an example of something that you struggled with and then overcame (or didn't!), giving an example of something you found particularly cool, or any other constructive way you can think of to participate.

Search entries or author Unread Subscribed

Reply

(https://santarosajc.instructure.com/courses/24402/users/74745)
James O'Hara
(https://santarosajc.instructure.com/courses/24402/users/74745)
Sep 1, 2017

Jim's Hopeful Helpful Tips for Assignment 3

Most important, read the assignment instructions carefully. Don't rush it. Re-read as you're designing, coding, and after coding. You'll save time in the long run.

Watch out for Style Convention 7p on the comparison overloads. An expression such as "a < b" has a boolean value of true or false. A function of type bool needs to return a boolean value, so "return a < b;" is correct. It isn't necessary to test the expression to decide what to return; the expression already has a boolean value.

wrong:

```
if (a < b)    // a < b is already true or false. doesn't need to be tested
    return true;
else
    return false;
```

Don't overlook the first bullet. Only one constructor function.

+=, -=, *=, /=, ++ and -- need to change the calling object. Except for post-fix ++ and --, these overloads do not need to create a new Fraction object. It would be an error if the += overload, for example, creates a new Fraction object with the value of the sum, and does not change the calling object to the value of the sum. (It's not wrong for += to create a new Fraction object, although it may be a little inefficient. What's wrong is failing to assign the sum of the calling object and the argument to the calling object.)

Notice the bullets that say there should be only one version for each +, -, *, and / overloads, and only one version for each of the comparison operator overloads.

Notice that the constructor should validate the denominator *parameter*, not the denominator data member.

Greg Fischer (https://santarosajc.instructure.com/courses/24402/users/15956)

(https://santarosajc.instructure.com/courses/24402/users/15956)

Sep 9, 2017

How much documentation is needed on these overloaded operators? For each one is it really necessary to write 15 words describing that the operator is overloaded? What would be an example.

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 9, 2017

That 15-word requirement is from Style Convention 1C. The S.C. for comments in classes is 1D. Also, you won't be graded on style issues for Assignment 3.

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 9, 2017

I'll do one for you, but keep in mind it's not required for assignment 3.

for operator+, something like this:

post: returns the sum of the parameters "left" and "right".

(Note precondition isn't listed because there is none.)

Dave

Kyle Stewart (https://santarosajc.instructure.com/courses/24402/users/85990)

(https://santarosajc.instructure.com/courses/24402/users/85990)

Sep 11, 2017

I had a weird problem where return (a < b); would always return true even when it wasn't, whereas if(a < b) Fixes the problem.

Anyway, I feel like the if statement not only looks nicer to read, but is also easier to edit if need be.

Just my thoughts, probably breaks another style convention knowing me.

Melissa Goh (https://santarosajc.instructure.com/courses/24402/users/63762)

(https://santarosajc.instructure.com/courses/24402/users/63762)

Sep 12, 2017

Hey Kyle, did you try

return a < b;

without the parentheses? I have no idea if that would make a difference.

Julia Otten (https://santarosajc.instructure.com/courses/24402/users/66384)
(https://santarosajc.instructure.com/courses/24402/users/66384)

Sep 13, 2017

These tips were quite helpful in the assignment, thank you for sharing them!

Hugh Lin (https://santarosajc.instructure.com/courses/24402/users/64934)
(https://santarosajc.instructure.com/courses/24402/users/64934)

Sep 13, 2017

I find this example code from the lesson quite similar to the example which you called wrong. What's the difference between them?

```
bool operator<(const feetInches& left, const feetInches& right)
{
        if (left.feet < right.feet){
            return true;
        }

        if (left.feet > right.feet){
            return false;
        }

        return left.inches < right.inches;
}
```

Alan Becker (https://santarosajc.instructure.com/courses/24402/users/110663)
(https://santarosajc.instructure.com/courses/24402/users/110663)

Sep 13, 2017

I believe the difference is that in the case of the lesson, we store both feet and inches, whereas in the assignment we store just fractions. If we were to store fractions as mixed numbers, then we'd run into a similar situation where we'd want to compare the whole numbers before then only comparing the fractions if the whole numbers are the same, but as it stands we can simply compare the fractions to one another directly.

Basically, the way it's implemented in the lesson allows for not bothering to compare the inches if the feet are different, because the one with more feet implicitly has more total inches regardless of how many extra inches are stored in the object.

Edited by [Alan Becker (https://santarosajc.instructure.com/courses/24402/users/110663)](https://santarosajc.instructure.com/courses/24402/users/110663) on Sep 13 at 11:19pm

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 14, 2017

Hugh, the difference is that the "wrong" example is

```
if (a < b)
    return true;
else
    return false;
```

whereas that example from the lesson would be something like

```
if (a < b)
    return true;
if (b < a)
    return false;
return c < d;
```

Not the same pattern, and no way to simplify the latter.

Reply

(https://santarosajc.instructure.com/courses/24402/users/114708)
Jesse Browne
(https://santarosajc.instructure.com/courses/24402/users/114708)

Sep 4, 2017

As I was starting assignment 3, I realized that you could use some operator overloads to implement the others. It is functional, but should it be done? For instance, I could use the '<' operator to implement the '>=' operator. Is this bad practice?

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 6, 2017

I would prefer to ***not*** use the < operator to implement <=, because the function is already so simple (a single line of code) and calling a function requires a lot of processing overhead during execution. Also, debugging could be a little more difficult, since you have to follow execution into a function.

There will certainly be times when I ***will*** use one operator to implement another one, in cases where it simplifies the code significantly. For an example, see the += operator in the feetInches class.

It's a judgement call. I certainly wouldn't take points of either way.

Reply

(https://santarosajc.instructure.com/courses/24402/users/118709)

Craig Nicholas (https://santarosajc.instructure.com/courses/24402/users/118709)

Sep 5, 2017

The Assignment 3 description says both "Fractions can be negative" and also "you may assume that all Fractions are positive. We'll fix that next week". I'm guessing the latter is an error, because client program and output have quite a few negative fractions. Is that right?

(https://santarosajc.instructure.com/courses/24402/users/74745)

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 6, 2017

This week you can write your code as if all Fractions are positive (even though they're not all positive). Next week you'll have to write code to deal with negative Fractions. So this week you can assume all Fractions are positive, even though Fractions can be negative.

Edited by James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745) on Sep 6 at 12:58am

(https://santarosajc.instructure.com/courses/24402/users/60154)

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 6, 2017

I see how that could be confusing, since the client program and output show a lot of negative fractions. The intention is that you shouldn't do any sort of special processing to make sure that negative fractions are handled correctly. You can see this in the output because you'll see negative denominators. Does that make sense.

(I believe the only place where this will make a difference is input and output functions.)

Reply

(https://santarosajc.instructure.com/courses/24402/users/73008)

Fred Straub (https://santarosajc.instructure.com/courses/24402/users/73008)

Sep 6, 2017

Do we want to write most of these arithmetic operation functions as friend functions?

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 6, 2017

The key is whether the left operand will always be a Fraction. If so, it can be a member function. If not, it has to be a friend. This is discussed in Lesson 16, which I suppose will be up soon.

Edited by ***James O'Hara** (https://santarosajc.instructure.com/courses/24402/users/74745)* *on Sep 6 at 5:11pm*

Reply

(https://santarosajc.instructure.com/courses/24402/users/59022)
Drew Rodrigues

(https://santarosajc.instructure.com/courses/24402/users/59022)

Sep 8, 2017

At first the requirement of "You should only use one function for each operator." confused me. I initially didn't think of friend functions and thought about how you would need to make multiple functions with the data types in certain orders, which would be inefficient. Just a hint for those of you who may be having the same issue.

Reply

(https://santarosajc.instructure.com/courses/24402/users/78465)
Sutter Laird

(https://santarosajc.instructure.com/courses/24402/users/78465)

Sep 8, 2017

I got really stuck for a long time in the RelationTest function because the comparisons kept throwing an "invalid operands to binary expression" error. I tested them in the main function and they worked fine, which greatly confused me. Finally, I realized that I had failed to grasp the importance of friend functions, and from there everything came together.

Reply

(https://santarosajc.instructure.com/courses/24402/users/118245)

Harry Doyle

(https://santarosajc.instructure.com/courses/24402/users/118245)

Sep 9, 2017

A couple of things.

1) Found the notes useful, I actually remember struggling with operator overloading with c++ way back in the day, notes and book were very helpful in that regard.

2) I did struggle with accessing the 2 data members (i named mine numerator and operator) at first. And in fact, to get around it, I made them public instead of private. That had more to do with I wasn't operating within the write scope (i believe is how you would term that). I went back, looked at the examples. Started simple (after I had written the entire program with the data members being public), and basically had to gut my program. Luckily, all the logic was basically the same, but all my function declarations and definitions needed to change. All and all, i learned something by banging my head against the wall for a minute and forcing myself to redo it correctly.

For example, original program looked like this:

```
public:
int i_numerator, i_denominator;

//no declaration for operator

Fraction operator + (const Fraction &f1, const Fraction &f2)
{
}
```

and i could access the numerator for both objects inside my method.

Correctly written (I believe) it looks like this:

*Edited by **David Harden** (https://santarosajc.instructure.com/courses/24402/users/60154) on Sep 9 at 1:13pm*

Eric Barnard
(https://santarosajc.instructure.com/courses/24402/users/68744)

(https://santarosajc.instructure.com/courses/24402/users/68744)

Sep 9, 2017

Looks like Dave edited your post. Careful about posting code on here, generally frowned upon.

Harry Doyle
(https://santarosajc.instructure.com/courses/24402/users/118245)

(https://santarosajc.instructure.com/courses/24402/users/118245)

Sep 9, 2017

Well, looks like my bad then. Was solely trying to articulate that at first the road I was going down while it worked, something felt wrong about accessing a public data member when all the examples had them as private and only accessible once you instance the object. Took a bit to figure out why (thus my code showing the difference). Live and learn. No code. Ok!

*Edited by **Harry Doyle** (https://santarosajc.instructure.com/courses/24402/users/118245) on Sep 9 at 2:45pm*

Reply

(https://santarosajc.instructure.com/courses/24402/users/81230) David Holstedt (https://santarosajc.instructure.com/courses/24402/users/81230)

Sep 9, 2017

Can we use the fraction class that we did in CS 10?

(https://santarosajc.instructure.com/courses/24402/users/60154) David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 9, 2017

Yes.

Reply

(https://santarosajc.instructure.com/courses/24402/users/76726) David Novosartov (https://santarosajc.instructure.com/courses/24402/users/76726)

Sep 9, 2017

I like this subject field of operator overloading because it gives you so much to play around with, pretty straight forward assignment where with each arithmetic operator you have to create functions for each one and call the numerators and denominators. Looking forward to adding more to this code with assignment 4!

Reply

(https://santarosajc.instructure.com/courses/24402/users/81230) David Holstedt (https://santarosajc.instructure.com/courses/24402/users/81230)

Sep 10, 2017

Do you need the class as a cpp and and header file, or just one file?

(https://santarosajc.instructure.com/courses/24402/users/81230)

David Holstedt
(https://santarosajc.instructure.com/courses/24402/users/81230)
Sep 10, 2017

Cancel that, I see we only need a single file.

Reply

(https://santarosajc.instructure.com/courses/24402/users/100398)
Tristan Girard
(https://santarosajc.instructure.com/courses/24402/users/100398)
Sep 10, 2017

I had the same problem as a few of you on here with the struggle to implement the friend function to allow the fraction to be evaluated. Lesson 16 cleared up most of the problems I had for implementation. And I appreciate all the questions you guys post to answer my same questions. I feel like I am not participating because I have no constructive questions, but I will post earlier with questions to help out more.

Reply

(https://santarosajc.instructure.com/courses/24402/users/114750)
Terrence Dubois
(https://santarosajc.instructure.com/courses/24402/users/114750)
Sep 10, 2017

So my code seems to be working well, but I wonder if there is a way to simply/condense my arithmetic + and - functions. Right now they create a temporary duplicate of the Fraction objects as well as a Fraction object titled "result," and do the operations on those temporary Fraction objects. This works, but I think it makes the function longer than it needs to be. I could take out the word const from the parameter, but that would mean having to move my data members to public.
*Edited by **Terrence Dubois (https://santarosajc.instructure.com/courses/24402/users/114750)** on Sep 10 at 1:24pm*

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)
Sep 10, 2017

Sounds like you have two local (temporary) Fraction objects. If so, yes, that's a little over complicated. It makes a lot of sense to have a "result" Fraction object, but the rest of the operations should involve ***accessing*** the two Fraction operands, but not ***modifying*** them, which is perfectly fine even if they are private and/or const.

Class, the following is a paragraph that you should spend some time on and make sure it makes sense, and if it doesn't make sense, ask some questions!

By the way, if you take out the word "const" from in front of your parameters, that won't mean having to move your data members to public. It's not related to making the data members public. Taking out the word "const" (and also the &) would actually make it so you could modify the operands. The reason not to do it is that it would be poor practice (objects should always be passed by reference).

One last word for students who have already written these functions and didn't use any local (temporary) Fraction objects at all in case you are worried that you did something wrong: your way is also a fine (maybe better) way to write the function. But most students don't do it that way.

Reply

(https://santarosajc.instructure.com/courses/24402/users/56692)
Anwar Polk
(https://santarosajc.instructure.com/courses/24402/users/56692)
Sep 10, 2017

I know it's probably just me, but I'm finding it difficult to connect my class's public functions with the rest of the client code. I'm running into a bunch of type-conversion errors. For instance, when I try to set my constant array Fraction fr[] equal to a whole number I get: "conversion from 'int' to non-scalar type 'Fraction' requested".

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)
Sep 10, 2017

Could you tell us the exact code from the client that you are referring to?

I'm guessing this means that you don't have a constructor that will take an int as a parameter.

(https://santarosajc.instructure.com/courses/24402/users/56692)
Anwar Polk
(https://santarosajc.instructure.com/courses/24402/users/56692)
Sep 10, 2017

Sure. I'm going to post the code that I wrote though, which I thought was kind of frowned upon.

It's an issue that I'm having in (what should be) the Fraction.cpp portion. I'm trying to get the fraction to allocate the integer arguments into their appropriate places (numerator and denominator):

```
Fraction::Fraction(const Fraction&)
{

numerator=0;
denominator=1;

for(int count=0; count < 3; count++)
{
Fraction fr[count];     <----------------------------------------------right here is where I'm having the error pop up
if(count = 0)
{
Fraction fr[count] = 0;
}
if(count > 0)
{
Fraction fr[count] = numerator;
Fraction fr[count + 1] = denominator;
}
}
}
```

(https://santarosajc.instructure.com/courses/24402/users/60154) David Harden

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 11, 2017

Anwar, please don't post code that is part of your solution that you are working out. That is absolutely frowned upon. What I asked you to do was tell us the exact code ***from the client.***

However, your code has enough work to do on it that I'll leave it up and try to make some observations about it.

(note: I am about to make some guesses about what you are trying to do here, but I could be totally off. In any case, we should take this discussion offline and communicate by email from here on.)

It appears that the heart of your error is that you are trying to access a variable (fr) that is declared in the client code, not in the class. When you are writing a member function such as this, you can't access client variables directly (otherwise the class would stop working if I changed the name of the variable in the class -- remember, these two pieces are being developed by different people at different times).

If this is in fact what you are trying to do, then the correct way to do this is to not use the variable (fr) that appears in the client code at all, but instead just refer to the calling object by using the data member "numerator" and "denominator"

Actually, looking at your code again, maybe I'm wrong about that, and you are trying to declare a new local variable named "fr". In that case, I would just say it doesn't make sense to have a function whose only task is to declare a local variable and give it a value. The value of fr would just be lost anyway once the function ends. Also, you are declaring an array named fr and putting "count" in the square brackets, but it is illegal to put a variable in the square brackets when declaring an array.

Finally, there's no need in this assignment to have a constructor with a Fraction object as the parameter at all.

Dave

Reply

(https://santarosajc.instructure.com/courses/24402/users/78834) Alex Dewey

(https://santarosajc.instructure.com/courses/24402/users/78834)

Sep 10, 2017

I can't seem to write an operator function that allows the fraction class to also take in integers :/ What should I be passing through that allows my function to do this?

Eric Barnard (https://santarosajc.instructure.com/courses/24402/users/68744) (https://santarosajc.instructure.com/courses/24402/users/68744)

Sep 10, 2017

Are you referring to creating a Fraction object with just 1 argument passed to it, like:

`Fraction(5)`

And having it create a Fraction of 5/1? Or are you referring to operator overloading and having ints used as an operand and not just Fractions?

Edited by **Eric Barnard (https://santarosajc.instructure.com/courses/24402/users/68744)** *on Sep 10 at 5:45pm*

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154) (https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 10, 2017

This sounds very similar to the problem Anwar is describing. I would suggest taking another close read of lesson 16.10 and 16.11. Let us know whether that helps.

Carolyn O'Rourke (https://santarosajc.instructure.com/courses/24402/users/102629) (https://santarosajc.instructure.com/courses/24402/users/102629)

Sep 10, 2017

Look at your constructor declaration and Dave's lecture notes pertaining to using a single constructor to handle all three cases of initialization. I hope this helps.

Alex Dewey (https://santarosajc.instructure.com/courses/24402/users/78834) (https://santarosajc.instructure.com/courses/24402/users/78834)

Sep 11, 2017

I'm posting this in case anyone had a similar problem. The solution lied in creating a function that would handle all inputs and also the use of the friend operator.

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154) (https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 11, 2017

Alex could you clarify this? I'm not sure what you mean.

Reply

(https://santarosajc.instructure.com/courses/24402/users/94493)
Jeryn Inman
(https://santarosajc.instructure.com/courses/24402/users/94493)

Sep 10, 2017

When we are required to follow SC for the next part of this assignment, will we need to put the six blank lines between the member functions of the implementation file? If that's explicit somewhere, my apologies but I don't see it.

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 10, 2017

The SC is "**Put at least 6 blank lines between function definitions.**" There's no restriction on which function definitions this should apply to and which ones it shouldn't apply to, so I think that means it should apply to all function definitions, whether they are member functions or not. Does that make sense?

(https://santarosajc.instructure.com/courses/24402/users/94493)
Jeryn Inman
(https://santarosajc.instructure.com/courses/24402/users/94493)

Sep 11, 2017

Yes, thanks for the clarification

Reply

(https://santarosajc.instructure.com/courses/24402/users/81860)
Eli Bowden
(https://santarosajc.instructure.com/courses/24402/users/81860)

Sep 10, 2017

Starting the assignment was a bit confusing but once i got a grasp on how overloading worked it went pretty quick. Lots of copy and paste!

Reply

(https://santarosajc.instructure.com/courses/24402/users/112937)
Lance Tran
(https://santarosajc.instructure.com/courses/24402/users/112937)

Sep 10, 2017

I'm a bit confused with the following program assignment. The assignment says that we will have to turn in a single file and next week we'll turn in three file parts? We're suppose to tweak the given client program, but is it suppose to compile and run? It says, "The class declaration will be first in the file, followed by the definitions of member functions, followed by the client code." Is adding a class to the client file all we are turning in?

Edited by ***Lance Tran (https://santarosajc.instructure.com/courses/24402/users/112937)*** *on Sep 10 at 6:52pm*

(https://santarosajc.instructure.com/courses/24402/users/74745)

James O'Hara
(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 10, 2017

No tweaking the client. The adjustments needed for next week's client file will be given, as this week's client file was. The adjustments you will be responsible for are the adjustments to the class to meet next week's additional requirements for the class.

Does that clarify?

Yes, adding a class to the client file is all we are turning in (along with sample run output and a comment on the submission page, indicating whether the code meets the requirements).

Edited by ***James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)*** *on Sep 10 at 8:05pm*

(https://santarosajc.instructure.com/courses/24402/users/112937)

Lance Tran
(https://santarosajc.instructure.com/courses/24402/users/112937)

Sep 10, 2017

Never mind, I finally got what the following assignment meant. In order for the program to work is to have a class named fraction (as followed), definitions for int variables, and set public and private specifiers. I had to set both numerators and denominators into obj files to make it executable. There was definitely a lot of copying and pasting when it came to setting assigning different operators (<, <=, >, >=, ==, !=), I thought that was absolutely tedious. I like how this type of program can be played with especially with assigning fractions to its given, coded arithmetics.

Reply

(https://santarosajc.instructure.com/courses/24402/users/56027)

Andrew Langwell
(https://santarosajc.instructure.com/courses/24402/users/56027)

Sep 10, 2017

I find making classes for client programs fun so this assignment was fun for me. Thanks James for your tip on returning boolean values in the > < == etc. operator overloads it enabled me to take out several lines of unnecessary if statement code.

(https://santarosajc.instructure.com/courses/24402/users/52648)

Sean Callahan
(https://santarosajc.instructure.com/courses/24402/users/52648)

Sep 11, 2017

Totally agree, Andrew! James' tips every week immensely help the completion of these assignments.

Style Convention 7p simplified each function to only a few lines.

Thanks Jim for the tips!

Reply

(https://santarosajc.instructure.com/courses/24402/users/63762)

Melissa Goh

(https://santarosajc.instructure.com/courses/24402/users/63762)

Sep 10, 2017

On this assignment, I was stuck for the longest time when overloading the += operator. I kept getting an error saying that "Overloaded operator must be a binary operator". I fixed it by adding "Fraction::" in front of "operator+=".

I found the lesson and discussion board being very helpful in completing this assignment.

Reply

(https://santarosajc.instructure.com/courses/24402/users/16215)

Don Barrett

(https://santarosajc.instructure.com/courses/24402/users/16215)

Sep 10, 2017

I keep getting this error that keeps the program from compiling.

error screencap.PNG (https://santarosajc.instructure.com/files/946882/download?download_frd=1&verifier=NUZbYeFvidpkXdxlJcHoSaPvUP1jqMRgrtLYhbQC)

(https://santarosajc.instructure.com/courses/24402/users/60154)

David Harden

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 11, 2017

Something's wrong with how you are setting up your project. Make sure you followed the steps I posted in the assignment 1 discussion:

- When you install Visual Studios Community, make sure you have Desktop development with C++ selected.

- The steps to creating an empty project seem a bit different in VSC 2017 than it was in VSC 2015. To create a new EMPTY project, go to File > New > Project... , then under Visual C++, select General, and create an Empty Project in there.

After that you can follow along with the Getting Started with Visual Studio tutorial starting at step 4.

Let us know how it goes.

Don Barrett (https://santarosajc.instructure.com/courses/24402/users/16215)
(https://santarosajc.instructure.com/courses/24402/users/16215)

Sep 11, 2017

I created a new project, selecting "Empty Project" and got the same result. Question: Am I supposed to have my class in a .cpp file, or a .h file?

error screencap 2.PNG (https://santarosajc.instructure.com/files/948374/download?download_frd=1&verifier=4e6n9HA3LVrd9Vuo0l4f2LJrWiBOOiTfHaJ3XEJ8)

Don Barrett (https://santarosajc.instructure.com/courses/24402/users/16215)
(https://santarosajc.instructure.com/courses/24402/users/16215)

Sep 11, 2017

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 11, 2017

Files that have a main() function in them should be .cpp files, not .h files. (And the assignment specifies that the given client will be in your file.)

Not sure what is going on. Just to confirm: you chose add -> new item to add your file to the project (step 5 of the tutorial), right?

Reply

(https://santarosajc.instructure.com/courses/24402/users/58611)
Harrison Finch
(https://santarosajc.instructure.com/courses/24402/users/58611)

Sep 11, 2017

Should we include a function to simplify our fractions?

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 11, 2017

No

Reply

(https://santarosajc.instructure.com/courses/24402/users/81230)
David Holstedt
(https://santarosajc.instructure.com/courses/24402/users/81230)

Sep 11, 2017

You say that we need to have the relational operators be handled with with just one function, but how can we do that if the function has to be accept different types? Is there a way to make a function not care what data type is inserted or something?

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 11, 2017

Yes. This is the subject of lesson 16.10 and 16.11. Please take another close look at those sections, and then let me know if that doesn't help.

Reply

(https://santarosajc.instructure.com/courses/24402/users/111918)
Neal Siegrist
(https://santarosajc.instructure.com/courses/24402/users/111918)

Sep 11, 2017

This assignment was a little overwhelming in the beginning due to the size of it. When I took it line by line it did not seem as hard. There were a few syntax situations that threw me off. For example, after having quite a few friend functions in a row I forgot the Fraction:: portion and I had my head spinning for 30 min trying to figure it out. Besides that it went pretty well and I feel like I am understanding classed really well and am excited to keep going on them!

Reply

(https://santarosajc.instructure.com/courses/24402/users/81230)
David Holstedt
(https://santarosajc.instructure.com/courses/24402/users/81230)

Sep 11, 2017

I got this error message:

```
prog.cpp:94:74: error: ‘Fraction& Fraction::operator*(const Fraction&, const Fraction&)’ must take eithe
r zero or one argument
 Fraction& Fraction::operator*(const Fraction& left, const Fraction& right)
                                                                          ^

Can you think of any reason I'd get that?
```

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 11, 2017

You are overloading the operator as a member function. It's a binary operator. So, if you're going to overload it as a member function, there should be one parameter, not two. Lesson 16.4 and 16.5 talk in detail about how many parameters an overloaded operator should have.

David Holstedt (https://santarosajc.instructure.com/courses/24402/users/81230)
(https://santarosajc.instructure.com/courses/24402/users/81230)

Sep 11, 2017

How do I overload it as a friend? I declared it with the friend keyword in the prototype.
*Edited by **David Holstedt (https://santarosajc.instructure.com/courses/24402/users/81230)** on Sep 11 at 7:33pm*

David Holstedt (https://santarosajc.instructure.com/courses/24402/users/81230)
(https://santarosajc.instructure.com/courses/24402/users/81230)

Sep 11, 2017

I couldn't figure it out from 16.4 or 16.5

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 11, 2017

This exact situation comes up in the feetInches class (except with +, instead of *). Could you take a look there and see how it's done? You could probably just follow the pattern of the final example and get it right, but you may also want to read through 16.10 and 16.11, perhaps later, to make sure you understand the concepts behind it.

EDIT: I realized that it may just be one detail that you missed. If you're trying to use a friend, then the function definition (the line of code you showed us in your error message) is not a member function, and so it shouldn't have Fraction::

Let us know how it goes.

David Holstedt (https://santarosajc.instructure.com/courses/24402/users/81230)
(https://santarosajc.instructure.com/courses/24402/users/81230)

Sep 11, 2017

Well, I got rid of the Fraction:: and got a new error message:

```
prog.cpp:95:11: error: ambiguating new declaration of ‘Fraction& operator*(const Fraction&, const Fra
ction&)’
 Fraction& operator*(const Fraction& left, const Fraction& right)
           ^~~~~~~~
```

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 11, 2017

For this one I would need to see your code. Could you send it to me?

David Holstedt (https://santarosajc.instructure.com/courses/24402/users/81230)
(https://santarosajc.instructure.com/courses/24402/users/81230)

Sep 11, 2017

I'm really frazzled right now, so I think I''m just going to call it for the night. It feels as though I'm almost there, but it's been that way for three hours, and I've made no progress.

Reply

(https://santarosajc.instructure.com/courses/24402/users/74662)
Josh Barnard
(https://santarosajc.instructure.com/courses/24402/users/74662)

Sep 11, 2017

I really enjoyed learning more about overloading operators and learning more about programming and C++. I think that being able to watch the lectures could be really useful for a class like this. I took a class during the summer that had all the lectures from the spring semester recorded and available on C3 for us to watch anytime. I think this would be really nice to have and help out with completing assignments on time.

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)
(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 12, 2017

Lots of videos available on the homepage in Canvas...did you miss them?

Reply

(https://santarosajc.instructure.com/courses/24402/users/50114)
Anthony Chui
(https://santarosajc.instructure.com/courses/24402/users/50114)
Sep 11, 2017

Are there any alternatives to friend functions that have the same result?

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)
Sep 12, 2017

You could always use a global function and then have the global function call a member function that actually does the accessing of the private data members. But that's a convoluted way of doing it...which is why friend functions are part of the language

Reply

(https://santarosajc.instructure.com/courses/24402/users/84454)
Angela Mao
(https://santarosajc.instructure.com/courses/24402/users/84454)
Sep 11, 2017

I was a little alarmed at first when the assignment said 20 functions, but I found all the practice was great. It was a little tricky for me to get the concept of overloading, but I think I am now fairly comfortable after all the repetition.

Reply

(https://santarosajc.instructure.com/courses/24402/users/81540)
Ryan Roncancio
(https://santarosajc.instructure.com/courses/24402/users/81540)
Sep 11, 2017

For the ++ and -- operators I was incrementing/decrementing the numerator by 1. The correct output file seems to show that it's supposed to increment/decrement the entire fraction by 1. The simplest solution I could think of was to continue to ++/-- the numerator but run it through a for loop where it would loop a number of times equal to the denominator. It seemed to work but I don't know if this is the most efficient way or if it will always work with every fraction.

Andrew Bieneman (https://santarosajc.instructure.com/courses/24402/users/99711)

(https://santarosajc.instructure.com/courses/24402/users/99711)

Sep 11, 2017

I think you can just increase/decrease the numerator by the denominator

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 12, 2017

From the assignment: To increment or decrement a Fraction means to add or subtract (respectively) one (1).

Reply

(https://santarosajc.instructure.com/courses/24402/users/4932) Kevin Drake

(https://santarosajc.instructure.com/courses/24402/users/4932)

Sep 12, 2017

The ++ and — **overloaded operators hung me up for a bit.**

Cinthya Rosales (https://santarosajc.instructure.com/courses/24402/users/110095)

(https://santarosajc.instructure.com/courses/24402/users/110095)

Sep 12, 2017

That one got me too. I kept making the mistake of incrementing the numerator by one instead of incrementing the whole fraction by one. Could have kicked myself once I realized my mistake.

Reply

(https://santarosajc.instructure.com/courses/24402/users/66085) Jeremy King

(https://santarosajc.instructure.com/courses/24402/users/66085)

Sep 12, 2017

This assignment was pretty straight forward once I made a couple of the functions. The biggest issue I had was determining which functions should be friend functions or not, but once I looked at the error messages it was pretty simple to correct.

Reply

(https://santarosajc.instructure.com/courses/24402/users/63243) Nathan Cortez

(https://santarosajc.instructure.com/courses/24402/users/63243)

Sep 12, 2017

I found that overloading operations to be really interesting as it gives you a lot of flexibility on what you could change to operators to become. I love the flexibility as if I wanted to make the addition operator display an output, I could really do that.

I know we weren't suppose to use the = operator as a member function, but I hope we'll be able to do that for the next assignment as I would love to be able to assign 'g = -1/3' immediately as a Fraction rather than having to use Fraction(1,3).. unless that's what we had to do for the assignment. I counted 20 member functions in mine so I wasn't particularly sure if that was suppose to be included or not. The repetitiveness of the assignment was a bit boring but either way it was nice to be able to try this out for each operators. It said not to use style conventions as we weren't graded on it so I didn't.

I've also been procrastinating lately and I'll admit to that, but I am having fun with this class. Trying to work on it :D

(https://santarosajc.instructure.com/courses/24402/users/60154) David Harden

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 13, 2017

We will be overloading = eventually (assignment 6, I believe), but it's impossible to make it so that we can assign a Fraction (or any other class) literal to a Fraction (or any other class) object, because the compiler simply doesn't recognize Fraction (or any other class) literals. To the compiler, -1/3 is simply -1 divided by 3, and there's no way to make it view it any other way.

Reply

(https://santarosajc.instructure.com/courses/24402/users/66384) Julia Otten

(https://santarosajc.instructure.com/courses/24402/users/66384)

Sep 13, 2017

I liked this assignment overall, I did get hung up for a bit working with the +=, -=, *= and /= operators but after writing it out on paper and working with the code for a bit I was able to get them to all work. I'm interested to start on the next part of this for this week's assignment and see what functionality we'll be adding to or changing about the code.

Reply

(https://santarosajc.instructure.com/courses/24402/users/4456)
Ryan Martino
(https://santarosajc.instructure.com/courses/24402/users/4456)
Sep 13, 2017

I feel like I understand the concept of overload functions but I'm having a hard time just getting it set up, I've never been supplied a main then had to write a class for it. I'm not sure if I add to the main or only write the class and its member functions? I emailed you in more detail dave.

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)
Sep 13, 2017

You shouldn't touch the client program at all. Just write the class and its member functions. You can start with just a constructor and an insertion operator, and that will give you enough to be able to test the "basicTest()" function in the client.

Let us know how it goes.

(https://santarosajc.instructure.com/courses/24402/users/57399)
Sam He
(https://santarosajc.instructure.com/courses/24402/users/57399)
Sep 13, 2017

Just write the Fraction class first, then read the client and make the functions step by step

Reply

(https://santarosajc.instructure.com/courses/24402/users/79900)
Peter Wright
(https://santarosajc.instructure.com/courses/24402/users/79900)
Sep 13, 2017

I got my class working, but I made too many functions which makes the code look somewhat disorderly.

Reply

(https://santarosajc.instructure.com/courses/24402/users/66861)

A.J. Hawkey
(https://santarosajc.instructure.com/courses/24402/users/66861)
Sep 13, 2017

I really enjoyed this lesson. I thought it was really neat to finally understand how all the operators worked. I can see how this can be useful in creating new types. My favorite part was thinking through how the post increment operator worked. It was a huge uhuh moment for me. I love those!!!

Reply

(https://santarosajc.instructure.com/courses/24402/users/4456)
Ryan Martino
(https://santarosajc.instructure.com/courses/24402/users/4456)
Sep 13, 2017

I've never used this assert() before I put it in the top of my constructor and included<cassert> but Im getting a assert failed? so I'm not quite sure what to do, since besides ask here. lol I'm getting the test fractions to come up just have zeros in the denom.

(https://santarosajc.instructure.com/courses/24402/users/4456)
Ryan Martino
(https://santarosajc.instructure.com/courses/24402/users/4456)
Sep 13, 2017

or if my default for the denom is set to 1, then is what you consider a 0 fraction being 0/(someother Int) in that case my code is fine and I can move on.

(https://santarosajc.instructure.com/courses/24402/users/64934)
Hugh Lin
(https://santarosajc.instructure.com/courses/24402/users/64934)
Sep 13, 2017

I'm also using the assert() and getting the assert failed error. Please help.

(https://santarosajc.instructure.com/courses/24402/users/64934)
Hugh Lin
(https://santarosajc.instructure.com/courses/24402/users/64934)
Sep 13, 2017

I figured it out. The reason for the error is that the default for the denominator member is set to 0. The error went away after I set the default to 1.
Edited by Hugh Lin (https://santarosajc.instructure.com/courses/24402/users/64934) on Sep 13 at 11:01pm

(https://santarosajc.instructure.com/courses/24402/users/60154)

David Harden

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 14, 2017

You asked, "is what you consider a 0 fraction being 0/(someother int)."

That is not just what I consider to be a 0 fraction. It's what ***defines*** a 0 fraction in mathematics.

Reply

(https://santarosajc.instructure.com/courses/24402/users/111342)

Matthew Young

(https://santarosajc.instructure.com/courses/24402/users/111342)

Sep 13, 2017

Really exciting seeing how common mathematical operators can be applied to work with our custom classes. This gives more control and ease of use in implementing custom classes. Still a bit confused by using the 'friend' syntax, but I'm sure over time I'll become more accustomed to it.

Reply

(https://santarosajc.instructure.com/courses/24402/users/110663)

Alan Becker

(https://santarosajc.instructure.com/courses/24402/users/110663)

Sep 13, 2017

I really enjoyed this week's subject material. Operator overloading seems like an extremely useful tool for working with classes moving forward. Previously numerous things about working with classes felt very restricted and just generally "off" to write code for, but now I feel a lot more free to implement just about any functionality that I could want to clean up my code and just generally make it "feel" better.

Reply

(https://santarosajc.instructure.com/courses/24402/users/89029)

Dany Rodriguez

(https://santarosajc.instructure.com/courses/24402/users/89029)

Sep 13, 2017