Overloaded Operators

Complete this worksheet by answering the questions using the given program. Make sure to color or **bold** your answers so they are distinguishable from the question.

Getting Started

- ☐ Make a copy of this document by selecting “Make a Copy” from the File menu.
- ☐ Go to https://replit.com/~ and log in with gmail using your Weber State email address.
- ☐ Open the starter code code linked below
- ☐ Click **fork**, located in the upper right hand corner, to create your own copy
- ☐ In your copy, go to the upper right hand corner, click **invite**. Then choose **Generate a Join Link**.
- ☐ Paste the generated link in the second bullet point below

Worksheet Program Links

- *Starter Code*: Repl Overloaded Operators
- *Finished Program Code*: https://replit.com/join/iwcolwxvju-trinanixon1

Part 1 - Overloaded Operators as Member Functions

1. Is + a binary or unary operator? Binary

2. Look in the Distance.cpp file under the Part 1 label. Observe the overloaded + operator definition. How many parameters does it have?
   It has one parameter, a Distance object that is const and represents the right hand side of the binary equation.
3. There are two sets of member variables being used in the definition, **`feet_`**, **`inches_`**, and **`rhs.feet_`**, **`rhs.inches_`**. Which set of member variables belongs to the implicit argument? Which set belongs to the explicit? The implicit is the one calling the operator, so that would be feet_ and inches_ which refer to the values of the Distance object. The explicit is what we are passing as a parameter to our function, so that would be rhs.feet_ and rhs.inches_ (rhs for right hand side of the equation).

4. There are three lines of code in the definition, what does each line of code do (Hint: write in order as pseudocode)?

   Step 1: Store in a new variable called ‘feet’ the result of adding feet_ (the value of the current object-implicit) and rhs.feet_ (explicit)
   Step 2: Same as above but with inches. Take the inches_ value and add it to the rhs.inches_ and store in a new variable called inches.

Step 3: create and return a new Distance object using the Distance constructor that takes 2 variables - feet and inches, which are values we got from the previous step 1 and 2, so they represent the sum of feet_ and rhs.feet_, etc.

5. Go to part 1 in main.cpp. Observe that there are three Distance objects that have been created: dist1, dist2, and dist3. What should the value of dist3 be if the result is printed in the following format:
   Should be:
   `feet_: 2 inches_: 13`

   Run the program to check your answer. Were you correct? No. It gave feet_: 3 and inches_: 1

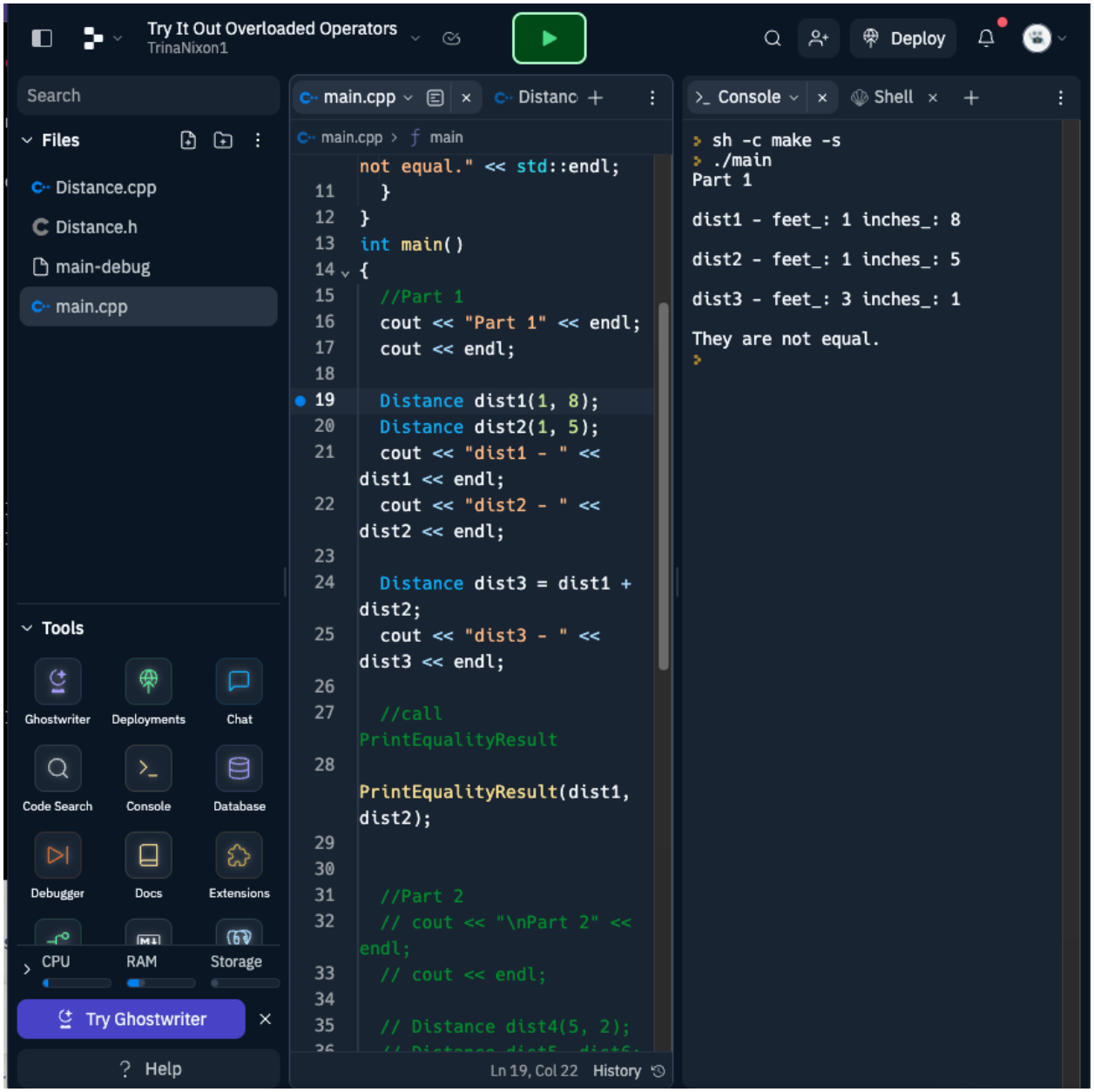
6. Observe the overloaded == operator. What does it do? (Hint: look at the definition in the Distance.cpp file.) The function takes a Distance object rhs as a parameter (this will be the rhs of the equality comparison), and returns true or false to indicate whether the 2 Distance objects are equal or not. It compares the feet_ and inches_ of the current object (left hand side) with the feet_ and inches_ of the rhs object (right hand side) using the == operator. Then it uses && to combine the 2 comparison results to make sure the entire expression is true. It will return true if the 2 Distance objects are equal and false if they're not.

7. Is it a binary or unary operator? It's a binary operator. It's checking for equality of 2 operands, a left-hand side and a right-hand side.

8. How many parameters does it have? It has one parameter, a Distance object rhs.

9. Write a function above main called `PrintEqualityResult` that accepts two `Distance` objects as parameters. The return type should be `void`. The function should check to see if the two given parameters are equal using the overloaded == operator. If they are, print "`They are equal!`". If not, print "`They are NOT equal!`".

10. Call the new function under Part 1 using the two Distance objects as parameters to check if it works correctly. Take a screenshot of your output and paste it below.



Part 2 - operator+ as a friend Function

1. Look at the operator+ in the Distance class under Part 2. Is it a friend function? How can you tell? Yes, because the friend keyword is being used in the function declaration

2. A friend function is not part of the class. Therefore, it needs to be sent to the object as a parameter. In this case, there are two parameters being sent, what are they? Distance const lhs, and float inches

3. Observe the function definition. What does it do? It adds a float value (inches) to a Distance object. It updates the inches_ variable of the lhs object with the addition result and returns a new Distance object with the updated inches value while keeping the feet

value of the original object.

It takes 2 parameters - Distance const lhs, which is the left hand side of the 2 operands, and float inches which is the # of inches to be added to the lhs object. It next adds the inches parameter that was passed in to the inches_ value (from the lhs object) then it creates and returns a new Distance object with updated values inches value which is used as the 2nd parameter for the Distance constructor while the feet_ value of the lhs object is used as the first parameter.

4. Go back to the main.cpp file and uncomment the code under the Part 2 label. What should the result stored in dist5 be? Run the program to check your answer. Were you correct?
   Yes, I was correct.
   Dist5 is the result of adding 15 inches to dist4, so that would make the result be 6 feet_: 6 inches_: 5.
5. In the definition, observe how the private variables are being used in the friend function. Notice that the instance of the object is used to access the member variables. Let's try an experiment. Remove the "`lhs`." from in front of the variable so it is just left with `inches_` instead of `lhs.inches_`. Run the program again. What error do you receive? Why do you think that error is there? We get error: use of undeclared identifier 'inches_'; did you mean 'inches'?
   The friend function won't recognize inches_ because it doesn't have access to the member variables.

6. Fix the code from the last comment to get the program working again. Then, add another operator+ as a friend function that does the same thing as the first, only this one should allow the user to enter the inches as the left operand.
   ```
   Distance operator+(float inches, Distance const rhs){
     inches += rhs.inches_;
     return Distance(rhs.feet_, inches);
   }
   ```
7. Use the new operator+ in main by adding 23 inches to dist4 and storing it in dist6. Run the program to make sure it works.

Part 3 - Extraction/Insertion Operators

1. We have been using the insertion << operator to display the Distance object. Notice the keyword **`friend`** before the operator declaration. Look below in the function definition. Is the keyword friend used there as well?Yes.

2. In the function definition, each of the Distance member variables are being streamed into an object using the << symbol. What is the name of that object? std::ostream&

3. What object is being returned from the function? The objects being returned is the reference to the std::ostream object. The object being returned is the same object being passed as a parameter.
4. Observe the Extraction operator >>. What are the parameters being used in this function? std:istream& in, Distance& rhs are the parameters. std::istream& in is a reference to the input stream from which data will be extracted. It allows the function to read data from the input stream. Distance& rhs is a reference to the Distance object. It uses this reference to store the extracted values for feet_ and inches_ into the member variables of the Distance object.

5. What is the name of the object that is being used to stream values into the Distance member variables?std::istream&

6. What is being returned from the function? A reference to the std::istream object that was passed as the first parameter.

7. Under Part 3 in main, use cin to populate the member variables of dist7 with 5 feet and 13 inches. Then use cout to print it. What is the output? Is it correct? Output is feet: 5 and inches: 13. No, not correct. It should be feet: 6 inches: 1.

8. In order to properly convert the inches, add a variable of type float called inches above the first statement in the operator>> definition. Replace d.inches_ with the new inches variable. Now call SetInches using the Distance object and use the new inches variable as its parameter.

9. Rerun the code in main using the values from earlier and verify that the program is producing the correct result. What should the result of printing dist7 be?
Yes, it correctly outputs feet: 6 inches: 1.

Verify that your program has all the necessary added code and make sure to include the link above before submitting this assignment.