# Program Statements

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 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, and click **invite**. Then choose to **Generate a Join Link**.~~
- ☑ ~~Paste the generated link in the second bullet point below~~

Worksheet Program Links

- *Starter Code*: Repl Program Statements
- *Finished Program Code*: https://replit.com/join/roysbgjvld-trinanixon1

## Example 1

Observe the statements below. Answer the questions and press "run" on the repl program to check your answers. We will assume that x is 5 and y is 5.

1. What is the order of operations in the statement below that will execute first? What do you predict the output will be?

```
cout << x + 2 * 5 + 1 << endl;
```

PEMDAS - so multiplication first (2 * 5). I predict the output would be 16.

2. What is the difference between this statement and that in the previous question? How does that change the output?

```
cout << (x + 2) * 5 + 1 << endl;
```

In the order of operations, parenthesis comes before multiplication so it would make the output 36.

3. What is the difference between the ++ being before the variable or after?

```
cout << x++ << " " << ++y << endl;
cout << x << " " << y << endl;
```

If the ++ is before the variable, this means it increments it first, then returns the value. If the ++ is after the variable, this means it is incremented by the value that precedes it. And the value returned is the value of the variable before it has been incremented.

4. Observe how the setprecision command works without the fixed notation being set. How many total numbers are printed?

```
cout << setprecision(3)  << 1.23456<< endl;
```

3 total numbers are printed;

5. Now observe the setprecision command with fixed notation. What is the difference? How many decimal places are printed?

```
cout << fixed << setprecision(3) << 1.23456 << endl;
```

3 decimal places only, are printed compared to without the fixed notation where it would be 3 total numbers.

6. Observe the setw command below; is the text right or left aligned within the setw space provided?

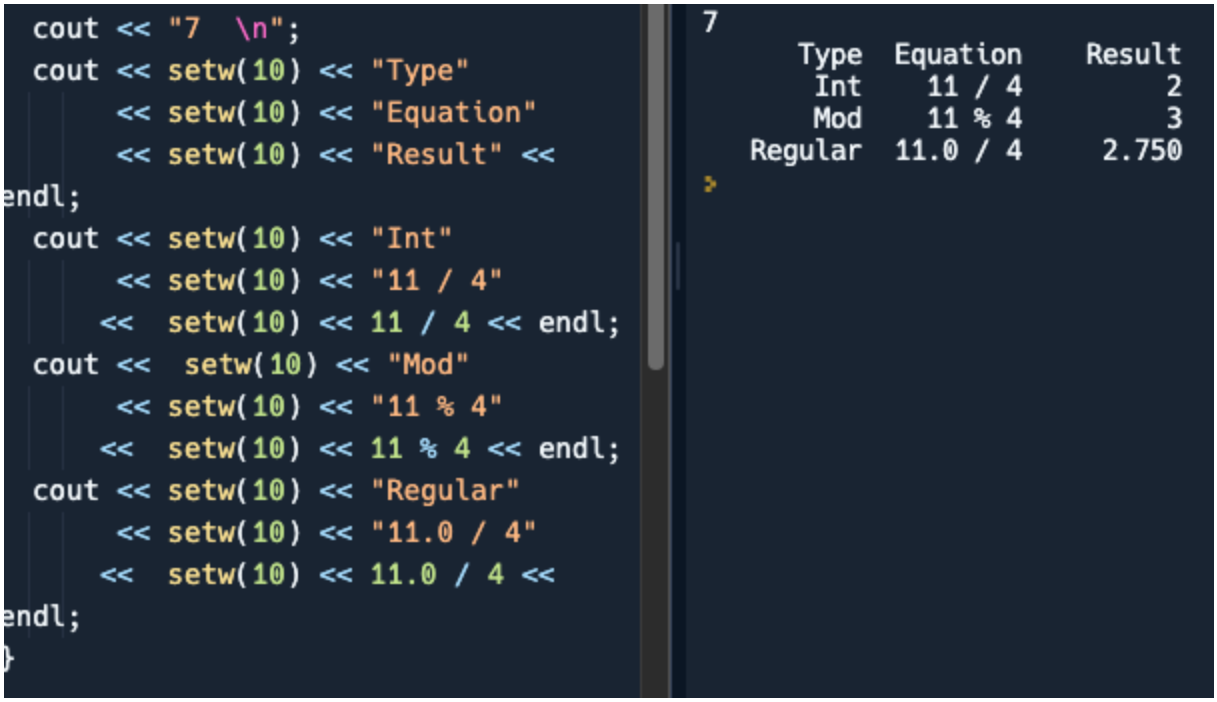
```
cout << setw(4) << 12  << setw(4) << 12 << endl;
cout << setw(4) << 123  << setw(4) << 123 << endl;
```

The text is right aligned.

7. Use the setprecision() and the setw() commands to display results similar to the following…

```
   Type   Equation    Result
    Int      11 / 4         2
    Mod      11 % 4         3
Regular   11.0 / 4      2.75
```

Paste a screenshot of your results:



```
  cout << "7  \n";
  cout << setw(10) << "Type"
       << setw(10) << "Equation"
       << setw(10) << "Result" <<
endl;
  cout << setw(10) << "Int"
       << setw(10) << "11 / 4"
      <<  setw(10) << 11 / 4 << endl;
  cout <<  setw(10) << "Mod"
       << setw(10) << "11 % 4"
      <<  setw(10) << 11 % 4 << endl;
  cout << setw(10) << "Regular"
       << setw(10) << "11.0 / 4"
      <<  setw(10) << 11.0 / 4 <<
endl;
}
```

```
7
      Type  Equation    Result
       Int    11 / 4         2
       Mod    11 % 4         3
   Regular  11.0 / 4     2.750
```

## Example 2

Look in the main.cpp file. Observe the bottom of the page in the main() function. There are three lines of code. You will notice that Example1() does not have a comment, but Example2() and example3() do. Modify the code to comment out Example1() and uncomment Example2().

Scroll up in the code until you find the method name Example2(). The code shows a partially working temperature calculator. Answer the following questions to complete the code, so it calculates the celsius correctly.

1. Run the program. Enter F to use Fahrenheit. Does the program allow you to type in the Fahrenheit or Celsius? Yes.

2. Observe the If statement under the comment that says //Determine the unit of measurement. Why do you suppose it used an or (||) instead of an and(&&)?
   Because it cannot be sure if the user will enter a lowercase or uppercase letter.

3. Continue with the program run and enter the value 50 for the Fahrenheit. Did the program calculate correctly? (You may need to look online to see the conversion of 50 from Fahrenheit to Celsius)
   No, did not calculate correctly.

4. Observe the equation under the "Do the Math" comment. Is there anything in the equation that would cause integer division? Fix the issue and make sure you get the

value 10. Yes, both are integers in the equation. If I make one a floating point number, it works correctly and gives me 10 as an answer.

5. Lookup on the internet what 60 degrees in Fahrenheit is in Celsius. Run the program using 60 degrees for Fahrenheit? Did it calculate correctly, and with the correct decimal places? Did not calculate correctly. There are no decimal places in the answer it gave.

6. Look at the variable declarations under the "Declare Variables" comment of the code. What data type is being used? Change it to something that allows for decimals, then run the program again to check it is working. The data type 'int' is being used. I changed it to 'double' and it works correctly.

7. Modify the "Display the Output" section to display to 3 decimal places.
   cout << fixed << setprecision(3) << "Celsius: " << cel << endl;
8. Your turn!! The code for Celsius to Fahrenheit is missing from the else statement. You may use the same variables, fah, and cel to make the program work if the user enters in the temperature in Celsius. Hint: Watch out for Integer Division.

```
else {
   //Get the input
   cout << "Enter Celsius: ";
   cin >> cel;

   //Do the math
   fah = (cel * (9.0/5)) + 32;

   //Display the output
    cout << fixed << setprecision(3) << "Fahrenheit: " << fah << endl;
 }
```