**App 1-5:**

I want you to get the 5 temperatures from the user and then the script calculates the average of the 5 values inputted from the user (so, it will be different each time).

**App 1-6:**

Same issue as App 1-5

Also, you are not outputting the commas between the numbers. It is a small detail but it is good practice because it increases the complexity of scripting the output statement (because you have to do much more switching between string output and variable output)

**App 1-7:**

You have a logical issue here. You have an embedded ***if*** condition that will always be FALSE because to get to that condition (temp > 80) you must pass the first condition (temp < 30) – and if you pass the first condition, you cannot pass the second.

if(maxtempToday < 30)

{

cat ("brrrrrrrrrr");

# the if statement is embedded in the previous if statement

if(maxtempToday >= 80) # this condition can never be met here!

{

cat ("enjoy the sunshine");

}

}

You have the exact same issue with your other ***if*** statement. I would first do this script without using the embedded ***if*** statements. You basically need four ***if*** statements. After you figure that out, I would challenge you to do the equivalent script with embedded ***if*** statements.

if(some condition)

{

}

if(some other condition)

{

}

Also, the curly brackets on lines 15 and 16 are not needed. There are not hurting anything (sort of like extra parentheses in an algebraic formula) but they are confusing because they hint that your code is sort of split at line 15 (and it is not)

**App 1-8 (part 1):**

**Keep your spacing consistent**

Lines 11 and 25-28 need to be adjusted. Yes, I am a tyrant about spacing – it is a life saver as your code gets longer and more complex.

**Small fix to make:**

If you type in ***-20***, you get no response from your script.

**And a challenge:**

Try to create an ***if-else-if*** structure with the same functionality that only has 1 check for an invalid statement (you currently have 2).

**App 1-8 (part 2):**

Works. Another wat to do the code:

if (grep(x=grade, pattern="a|A"))

{

cat ( "90-100");

}

I only point this out to whet the appetite for ***grep()***, which is incredibly powerful. ***grep()*** is the answer to the question: How do I deal with text input where people are not consistent with how they type things in?

**App 1-9:**

I want you to rethink this script and put all the ***readline()*** inputs from the user at the top of the script. After the script gets the inputs, then it does the checks on the inputs.

In general, when you design a script it is best to put the variables you are inputting (whether from a user, database, csv file, or another package) at the top of your script. The variables are like resources for your script and it is easier to read and tweak code if one can see all the resources the code needs at the top of the script.

This is sometimes impossible but in this it is not.

Also, it looks like you have two scripts, one starting at line 1 and the second at line 47. You should only have one. I know that the typical R programmer breaks scripts up into blocks that they individually execute. I would love to break R programmers of this habit – it makes code harder to share and debug.