1. One notable obstacle I had to overcome was returning the correct number of badBeats. Many times, my tests were off by 1 due to me not adding 1 or adding 1 when I should not have to “i” within the for loop when referencing certain cases where badBeats should be set to currentBeats and this caused badBeats to be a different value than expected. I solved this obstacle by tracing through my code and finding where exactly I needed to add or subtract 1. Another notable obstacle was having to call a switch statement to recognize each color as valid; I solved this issue by creating another function isColor() to test whether or not the char passed was a valid color.
2. **For hasProperSyntax:**

Repeatedly iterate through tune

Return false if the character is not a valid color digit or slash

If the character is a color, first return false if the next index is out of bounds

Check if the character past the color is a digit

Return false if the next index would be out of bounds

Check if the character past the first digit is a ‘/’

If there is a second digit past the first digit

Return false if the next index would be out of bounds

Check if the character past the second digit is ‘/’

If not, return false

Check if the character is just a ‘/’ with no color attached to it

Return true if the code successfully does not return false before breaking from the loop

**For convertTo:**

Return 1 immediately if tune doesn’t have proper syntax

Initialize the current beat count to 0 and declare a new string

Repeatedly iterate through tune

+1 to beat count if the char is a slash and append “x” to new string

If not, and the character is a color

+1 to beat count

Check if the char past color is 0 or 1 and that there’s no digit after that

If so set badBeat equal to beat count and return 3

Check for slash after the color

There is only 1 beat of that color so +1 to beat count and add lowercase letter representing color to new string

Check if the char after the color is a digit

Check if there are 1 or 2 digits and convert from string to int n

Repeatedly iterate through the slashes past the last digit n times

Ensure indexes stay in range or set badBeat to beat count

and return 4

If char found is not a slash, set badBeat to beat count and

return 2

If not, assume char found is slash and +1 to beat count

If the loop runs through, append n uppercase characters

representing the color to the new string

Set instructions to the placeholder string and return 0 once outer loop runs through

3.**PROPER SYNTAX**

* assert(hasProperSyntax("")); ***- checks if empty string will pass as proper syntax***
* assert(hasProperSyntax("G/")); ***- checks if uppercase will pass as proper syntax***
* assert(hasProperSyntax("r//Y/g3///o/")); ***- checks if single digit with correct slashes past will pass as proper syntax***
* assert(hasProperSyntax("y03///r10//////////")); ***- check if 2 digits with correct slashes past will pass as proper syntax***
* assert(hasProperSyntax("///")); ***- checks if slashes only pass as proper syntax***
* assert(hasProperSyntax("g/b//")); ***- checks if single slash after 2 colors passes as proper syntax***
* assert( ! hasProperSyntax("g/z//")); ***- checks if incorrect color will fail as proper syntax***
* assert(hasProperSyntax("r/")); ***- checks if lowercase will pass as proper syntax***
* assert(!hasProperSyntax("r3")); - ***checks if color + digit with no slashes after will fail as proper syntax***
* assert( ! hasProperSyntax("r")); ***- checks if color with no digit or slash after fails as proper syntax***

**CONVERT**

badb = -999; // so we can detect whether this gets changed -

* assert(convertTune("r//g/", instrs, badb) == 0 && instrs == "rxg" && badb == -999); ***- checks if convertTune will return 0 and instrs will be changed from a convertible tune to its correct conversion of “rgx” and leave badb unchanged***

instrs = "WOW"; // so we can detect whether this gets changed -

badb = -999; // so we can detect whether this gets changed

* assert(convertTune("r", instrs, badb) == 1 && instrs == "WOW" && badb == -999); ***- checks if convertTune will return 1 and both badb and instrs remains unchanged as the tune passed is not proper syntax***
* assert(convertTune("r/y3//g/r/", instrs, badb) == 2 && instrs == "WOW" && badb == 4); ***- checks if the code will check if there is the incorrect number of slashes and thus return 2 while setting badb to the beat of issue***
* assert(convertTune("r/g03///R//B11///////////", instrs, badb) == 0 && instrs == "rGGGrxBBBBBBBBBBB"); ***- checks if the code will correctly return 0 and convert the tune with double digits (even if the first digit is 0) to intrs***
* assert(convertTune("r/b/r13/", instrs, badb) == 4 && badb == 4); ***- checks if the code tests for prematurely ending code that is out of bounds and returns 4 and sets the correct beat to badb on a tune with double digits***
* assert(convertTune("r/b/r3//", instrs, badb) == 4 && badb == 5); ***-checks if the code tests for prematurely ending code that is out of bounds and returns 4 and sets the correct beat to badb on a tune with single digits***
* assert(convertTune("r1/", instrs, badb) == 3);  ***- checks if the code checks for incorrect digit usage of 1 or 0 that is inconvertible and returns 3.***