1. The obstacles I overcame were

a) To understand what is “well-formed” and what is “playable”

b) How to get three variables for translateNote

c) How to determine if “[]” is needed

d) How to find the problem when the output is wrong

2. isABCDEFG:

if the character is either A, B, C, D, E, F or G

return true

otherwise

return false

translateNumbers:

if the character is ‘1’

return 1

if the character is ‘2’

return 2

if the character is ‘3’

return 3

if the character is ‘4’

return 4

if the character is ‘5’

return 5

if the character is ‘6’

return 6

otherwise

return 0

isWellFormed:

if input is empty

return true

if input doesn’t end with a slash

return false

if input doesn’t start with a slash or one of the letters from A to G

return false

repeatedly for every character in the middle:

if the character is one of the letters from A to G

or if the character is ‘#’s or ‘b’s following one of the letters from A to G

or if the character is digits following one of the letters from A to G or following either ‘#’ or ‘b’

or if the character is slashes

return true

otherwise

return false

isPlayable:

repeatedly for every character in the string:

if the character is a digit and

if the digit is 2,3,4 or 5 except it’s a 2 following Cb

or if the character is a 6 following C

or if the character is a 6 following Cb

or if the character is a 1 following B#

return true

otherwise

record the position of the first unplayable note

return false

translateTune:

repeatedly for every character of the string:

if the character is a letter

increment the number of letter

if the character is a slash and

if the number of letter if larger than 1

store a ‘1’ in the string bracket

else

store a ‘0’ in the string bracket

reset j to 0

if the tune is well-formed and is playable

clear string instruction

if the first character is a slash

add a space character

increment counter k

increment counter h

repeatedly as k is smaller than the size of string tune:

if the character of bracket is ‘1’

add first half of bracket to instruction

a beat with a bracket starts

if the character is a slash and a beat with a bracket has been started

add the other half of bracket

the beat ends

and if the former character is a slash

add a space character to string instruction

increment counter k

otherwise initialize octave to 4

initialize accidentalSignal

set the character to noteLetter

if the next character is a digit

set octave to the digit of next character

increment k twice

or if the next character is either ‘#’ or ‘b’ and the character after the character is a digit

set accidentalSignal to the next character

set octave to the digit of the character after next character

add translated character to instruction

increment k three times

or if the character is either ‘#’ or ‘b’ but is not followed by a digit

set accidentalSignal to the next character

add translated character to instruction

increment k twice

otherwise

add translated character to instruction

increment k

return 0

or if the tune is not well-formed

return 1

otherwise

initialize badBeat (the number of beat) to 1

repeatedly for every character in string tune:

if the note is the first unplayable note

break the loop

if the character is slash

increment badBeat

return 2

3. isWellFormed:

empty beat well-formed

#A/ wrong order, not well-formed

#3/ no letter, not well-formed

A not well-formed

$#% not well-formed

translateTune:

A/ single letter

AB/ more than one letter

A3/ a letter followed by a digit

G1/ well-formed but not playable

G3B3DD5//G/A1/A3B/C5/B3D5// input with single unplayable tune

G3B3D#9D5//G/A1/A3B6/C5/B3D5// input with multiple unplayable tunes

G3B#3DD5//G/Ab/A3B/C5/B3D5// multiple beats with ‘#’s , ’b’s and empty beats