Maeneka Grewal – 204901948

CS31

Project 3 Report

a. In terms of the skeleton of this code, I did not have many problems. However, after I had coded a general skeleton of the program that ran successfully, I encountered a great deal of difficulty in altering the code in small ways to account for the specifics of the program. For example, my original skeleton did not include a case for when the beat is empty. It took me many tries to place the correct statement that accounted for the empty beat, just because it was so difficult to go back and trace through all the if’s and for-loops after they were already written. This also occurred when I realized I needed to group together multi-note beats in brackets. However, I eventually commented out my code in detail so that it was easier to trace through the if’s and for-loops, allowing me to insert small changes quicker.

b.

hasCorrectSyntax- accepts a song string

first checks to see if the song is empty or is just a slash and returns true

then checks to see if the song ends in a / like it should, returns false if it doesn’t

begins to breakdown the song into a series of beats using a for loop

for (traverse song)

if the character is not a slash

add the character to the substring

if the character is a slash

the beat is finished, send it to be evaluated for syntax

if the syntax is incorrect, return false

after the whole song is traversed and false was never returned, return true meaning no beats were syntactically incorrect

evaluateBeat- accept a string beat

first checks to see if the beat is empty or is a space and returns ture

for loop used to check each character of the beat

if the character isn’t a digit, letter, or sign (separate functions used to check this) returns false

if the character is a sign and is the first character of the beat, or the previous character is not a letter, returns false

if the character is a digit and is the first character of the beat, or the previous character is also a digit, returns false

after traversing the whole beat and nothing was returned, returns true meaning the whole beat was syntactically correct

encodeSong- accept a song string, and changeable instructions and badbeat

first sets a string ‘original’ to instructions, incase this needs to be recovered later

sends the entire song string to hasCorrectSyntax, and if there is a syntax error, returns 1 and exits the function with badBeat and instructions unchanged

sets instructions to empty, a string called note to empty, a string called beat to empty, and a string called newNote to empty

initializes a counter called beatCount to 0

for loop to traverse through song and break into beats

if the character is not a /, it is added to the beat

otherwise the character is a /, the beat is complete and can be broken into notes

newNote is reset to “” with each pass of the for loop and beatCount is incremented

check to see if the beat is empty or just a space, and if so, add either one or two spaces to instructions respectively

for loop to traverse through beat and break it into notes

if the note is empty and the character is a letter, add it to the note

if the character is not a letter, meaning it’s a sign or digit, it should also be added to the note

if the next character is a letter, this is the end of the note

initialize variables octave, letter, and sign

send the note to be evaluated for playablity, while also changing octave, letter, and sign to what they should be

if the note is playable, encode the note and add it to new note, and reset note

otherwise the note is not playable, so badBeat is set the beatCount, instructions is reset to original, and 2 is returned

if newNote has multiple notes, meaning the beat contains multiple notes, add it to instructions with brackets around it

otherwise just add it to instructions

reset the beat to empty

the whole song has been traversed and encoded properly and added to instructions, so return 0

evaluateNote- accepts a note, changes the octave, letter, and sign

the maximum length of a note is 3, so check to see each position of each character and make sure that the note is playable according to spec in series of if statements

if the note is one character long, this must be just a letter so set the octave to 4, the letter to the appropriate letter, and the sign remains empty. Return true

if the note is 2 characters long, it is either a letter and a sign or a letter and a digit. Check to make sure the digit is in the appropriate octave range. Set the values of letter, octave, and sign to what they should be

If the note is 3 characters long, it must be a letter, a sign, and then an octave. Once again, the octave must be in the right range. Set the values to what they should be.

Account for strange cases in the spec through separate if statements

c.

|  |  |
| --- | --- |
| Reason | Input |
| Empty string, which is zero beats and syntactically correct and so encodeSong returns 0, instructions should also be set to empty, and badBeat is unchanged | “” |
| A beat with no notes so it is syntactically correct and instructions should just be a space, encodeSong returns 0, and badBeat is unchanged | / |
| A beat with a rest that is syntactically correct, and instructions becomes a double space, encodeSong returns 0, and badBeat is unchanged | / |
| A simple song of one note that is syntactically correct, encodeSong returns 0, badBeat is unchanged, and instructions becomes Q | A/ |
| A simple song of one note that is syntactically correct, encodeSong returns 0, badBeat is unchanged, and instructions becomes F | Cb/ |
| A simple song of one note that is syntactically correct, encodeSong returns 0, badBeat is unchanged, and instructions becomes A | E#3/ |
| A multinote beat that is syntactically correct, encodeSong returns 0, badBeat is unchanged, and instructions becomes **[$KR3]** | AbE#D5F#2/ |
| A multibeat song that is syntactically correct, includes multinote beats, spaces, etc, returns 0, badBeat is unchanged, and instructions becomes **[SFHR] LQ[DW]E[FR] L L [GT] ERT\*[FU] L L** | G3B3DD5//G/A/A3B/C5/B3D5//G//G//CE5//C5/D5/E5/F#5/B3G5//G//G/ |
| A multibeat song that is syntactically correct and includes multinote beats, but has an unplayable note so it returns 2, instructions is unchanged, and badBeat returns the bad beat number which is 5 | C3/F4/ /G#2/C9//E4/ |
| A song that has a syntax error, so instructions is unchanged, bad beat is unchanged, encodeSong returns 1 | Cb4/Z/E4/GDD3/ |