Huimin(Hermmy) Wang

Dr. Smallberg

CS 31 Lec 2

30 October 2017

Report – Project 3

Notable Obstacle

The hardest part of project 3 for me is inserting and appending characters to a string at the correct index, more specifically, adding the square bracket to the beat which contains more than one note. I overcame the obstacle by counting the number of uppercase letters in a beat and resetting the counter after a slash. I also researched the functions built in the string libraries that allow me to insert a character in and append it to a string.

Pseudocode

Check if a song string has correct syntax

If a song string starts with a digit or an accidental sign, or is nonempty but does not end with a slash

Return false

If a song string is an empty string

Return true

Repeatedly:

Find any uppercase letter higher than G

Return false

Find any invalid characters besides uppercase letters, digits, accidental signs, or slashes

Return false

Repeatedly:

Find a note letter that is followed by a note letter in a song string

Jump to the next note letter

Find a digit not followed by a note letter or a slash, or an accidental sign not followed by a note letter, a digit, or a slash, or a slash not followed by a note letter or a slash

Return false

Return true

Encode the notes in the song string

If the octave is greater than 9

The note is not playable

Switch the note letters from ‘A’ to ‘G’

Assign an index to each of the note letters

If a note letter is outside the range

The note is unplayable

Switch the accidental signs

‘#’ moves the index one place up

‘b’ moves the index one place down

‘ ’ does not move the index

The note is unplayable in any other cases

Set the sequence number to the index of a key map

If the sequence number is outside the range of the key map

The note is unplayable

Set the instruction to the character on the key map at the index of the sequence number

Encode the song string into a set of instruction

Set the original instruction to an initialized instruction

If the song string has correct syntax

Repeatedly

Find every note letter

Increment the number of notes

Call encodeNote to encode notes with no digit character, with a digit character, with an accidental sign followed by a digit, and with an accidental sign only

concatenate the instruction

if the instruction is a blank space

Set the bad beat to the current beat

Leave the instruction unchanged as the initialized instruction

Return 2

Find every beat

If the beat has more than one note

Insert a square bracket at the beginning of the beat

Append a square bracket at the end of the beat

Reset the number of notes of the next beat

Increment the number of beats

Find every empty beat

Add a blank space to the instruction

Return 0

Else the song string does not have correct syntax

Return 1

Test Data

* “D5//D/”: a song string that has an empty beat. My program originally did not handle this correctly because I did not include the statement that handles a double-slash.
* “D5//Z/”: a song string with a note letter that does not have the correct syntax.
* “A/Bb/C3/F#3//ABbC3F#3/”: a song string with a single note letter, a note letter with an accidental sign, a note letter with a digit, a note letter with an accidental sign and a digit, and a beat with more than one note. My program originally did not handle this song string because the bracket was placed in the wrong position.
* “”: an empty song string.
* “/”: a song string with only one beat with no notes.
* “C/F9G/”: a song string that has the correct syntax but is not playable.
* “C/F4G//@/”: a song string that contains an invalid character.