Raymond Lin

Smallberg

CS31, Disc. 1D

30 October 2017

Project 3 Report

2. a) There were many obstacles that I encountered during this project. First of all, I noticed it was challenging simply to read and fully understand the project based on the spec. I took more than an hour just reading the spec and understanding what was required. Additionally, I found it challenging to implement the logic behind the two test cases - to see if the song string was valid and to see if the song string was playable.

b) For the ‘hasCorrectSyntax’, I designed the function such that it loops through the entire song string and tests each character to see if it is valid. At the end, I tested to see if the last character was a ‘/’ or a ‘ ‘.

//test every character in song to see if they are valid; if there is an invalid combination, return false  
bool hasCorrectSyntax(pass in the song string)

{

for (loop through the entire song string) {

switch (note) {

//Cases for valid characters, return true

default:

//if not valid, return false

}

}

if (check to see if last character is a ‘/’) {

//if it is not, return false

}

}  
  
For the ‘encodeSong’, I designed it by testing many cases. First, I checked if the syntax of song tring was correct. If it was not, the program returns one. If the syntax is correct, I tested to see if the song was playable. There were many cases, for example, number characters cannot be the first character of the string, and octaves 0, 1 and 6 were impossible to be played (unless if 6 was the note C). If all these conditions are passed, the instructions string was set to the song string, and the program returned 0. If it failed, the bad beat was recorded and the program returned 2.  
  
int encodeSong(song string, instruction string pointer, badbeat int pointer)

{

//if has right syntax, check if playable, else return 1

//if song is empty string, return 0

if (syntax is correct) {

//check to see if song is playable

for (loop through the song string) {

switch (character at the position) {

//if it is a slash, plus a beat. It is also playable. If it is next to another slash, encode it as a ‘ ‘

//if it is a space, it is playable.

//if it is a valid note letter, it is playable. If there is a note letter after it or a slash, encode it.

//if it is an accidental sign, and if the character after is not a digit and the character before it is a note letter, encode it.

//if it is a 1, the character before must be a # and a ‘B’ before that

//if it is a 2, if the character before it is a note letter, encode it. If the character before is an accidental sign and the one before that is a ‘C’, encode it. If the note is Cb2, it is unplayable.

//if it is 3, 4, 5, if the character before is a note, or the character before is an accidental sign and the one before that is a note letter, encode it.

//if it is 6, if the character before is a ‘C’, or if the character is a ‘b’, and the one before is ‘C’, encode it.

//default case: the character is unplayable

}

} //end of for loop

if (song is playable) {

if(song has at least one note)

//set instructions to the song string

Return 0

}

}

else { //if the string has the wrong syntax

//Return 1

}

}

c) Here is the table for the test cases:

|  |  |
| --- | --- |
| Song String | Reason |
|  | Empty string is a valid song |
| /// | Song with no notes is valid |
| G/ | Single beat song, without accidental signs |
| G#/ | Single beat song, with an accidental sign |
| AG#/ | Single beat song, with two notes and an accidental sign |
| A#G#/ | Single beat song, with two notes and two accidental signs |
| A#3G#/ | Single beat song, with two notes, two accidental signs and an octave number |
| A#3G#3/ | Single beat song with two notes, two accidental signs and two octave numbers |
| A/E/D/A/ | Multiple beat song, each with one note |
| AE//E//FAD/ACE/ | Multiple beat song, each with multiple notes |
| AC#E//Eb//F#AD/AbC#E/ | Multiple beat song, each with multiple notes and multiple accidental signs |
| A3C#E//E//F#3A3D4/A3C#E/ | Multiple beat song, each with multiple notes, multiple accidental signs, and multiple octave numbers |
| C0C0DC0DD/EFBC/B9/ | Octaves are unplayable, because octave 0 and 9 does not exist on the keyboard |
| CCDCDD/#E#FbB#Cb/bB#/ | Accidental signs are unplayable, because it is at the beginning of a beat |
| CCDCDD/3E#FbB#Cb/4B#/ | Octaves are unplayable because it is at the beginning of a beat |
| CCDCDD/E#FbBb#Cb/B#/ | Accidental signs are unplayable because there are two in a row |
| CCDCDD/E#Fb34B#Cb/B#/ | Octaves are unplayable because there are two in a row |
| CCDCb2DD/E#FbB#Cb/B#/ | Cb2 is unplayable |
| CCDC#6DD/E#FbB#Cb/B#/ | C#6 is unplayable |
| A3C#E//E//F#3A3D4/A3Cb6/ | Cb6 is playable |
| A3C#E//E//F#3A3D4/A3C6/ | C6 is playable |
| A3C#E//E//B#1A3D4/A3C#E/ | B#1 is playable |