Computer Science Project 3

# Overcoming obstacles

Initially, I had problem separating and checking the validity of each musical note. But then I figured out that by keeping track of the position or index in the string, I could easily check the validity of the characters and a particular combination of them. I also had difficulty translating a set of musical notes in a beat and placing the translated characters inside a square bracket. I overcame this problem by declaring a temporary string that held the translations of one beat at a time. This temporary string was then added to the final translation string along with square brackets after the end of each beat. Of course, I also took caution at this point to avoid placing square brackets if zero or one note was present in the beat.

# Program Design:

* isTuneWellFormed function: The function checks the validity of a tune by checking if a character and up to 2 of its succeeding characters are valid or not. There are multiple checks using if conditions to ensure a valid sequence of characters is present. If none of these if conditions are satisfied at a particular point, the function returns false. Once all the characters are checked for validity, the last character is ensured to be a ‘/’ character before returning true.
* translateTune function: This function translates the tune by using a temporary string and a final string. The temporary string holds all the translations in one particular beat. Each individual note translation is done by the function translateNote, and the return value is added to the temporary string. Once a beat is complete, the temporary string is added to the final string (with square brackets if necessary) and the temporary string is reset as an empty string. If the tune is not well formed, the program doesn’t attempt to translate the tune and returns 1. If the translation is successful, the final string is stores in the instructions parameter, and the function returns 0. Finally, if a non-playable tune is present, the function returns 2 and the badBeat parameter is given the value of the no. of completed beats plus one.

# Pseudocodes:

**isTuneWellFormed Function**

*position in string is 0*

*repeatedly:*

*check if character is a ‘/’*

*+1 to position in string*

*otherwise*

*check if character is a letter between A and G inclusive*

*check if letter is followed by an accidental*

*check if accidental is followed by number*

*+3 to position in string*

*otherwise*

*+2 to position in string*

*check if letter is followed by number*

*+2 to position in string*

*otherwise*

*+1 to position in string*

*check if the position of the string was NOT changed*

*return false*

*end loop*

*if the string is empty or last character is a ‘/’*

*return true*

*otherwise*

*return false*

**translateTune Function**

*initiate variables letter, octave (=4) and accidental (=space character)*

*check if tune is well formed*

*repeatedly:*

*check if character is ‘/’*

*Add* translation of beat *string to* translation of tune *string and put square brackets if necessary*

*reset* translation of beat *string as empty string*

*check if character is a letter between A and G inclusive*

*record value in letter*

*check if letter is followed by an accidental*

*record value in accidental*

*check if accidental is followed by number*

*record numerical value in octave*

*otherwise*

*record 4 in octave*

*check if letter is followed by number*

*record numerical value in octave*

*record space character in accidental*

*check if* translateNote *function with parameters octave, letter and accidental outputs space character*

*store value of badBeat as the current beat number + 1*

*return 2*

*add return value of* translateNote *function to the* translation of beat *string*

*reset octave to 4 and accidental to space character*

*end loop*

*store* translation of tune *string* *in instructions*

*return 0*

*end if condition*

*return 1*

# Test Data:

The following test cases were used to test the functions in the program, and all these test cases were appropriately handled.

|  |  |  |
| --- | --- | --- |
| **Return value** | **Test Cases** | **Reason/Purpose** |
| 0 | No input | To ensure the translation is an empty string |
| Sequence of '/' | To ensure the translation is a sequence of space characters |
| Beats with characters A, B, C, D, E, F and G | These inputs must provide a valid translation |
| Beats with note letters followed by '#' or 'b' |
| Beats with note letters or accidentals followed by 2, 3, 4 or 5 (Except Cb2) |
| Extreme values: B#1 or C2 and B#5 or C6 | Boundary values may not be processed correctly if there is an error |
| A combination of any of the above cases | To ensure that the program translates correctly given a valid input |
| 1 | A space character | Space characters are not valid inputs |
| Beat doesn’t end with '/' | Every beat must end with a '/' character |
| Beats with alphabetic characters after G | Only note letters from A to G are acceptable |
| A numerical digit not preceded by a note letter or an accidental | These combinations do not match the syntax of musical notes |
| An accidental not preceded by a note letter |
| Any other unacceptable characters such as punctuations |
| 2 | The note Cb2 and a note belonging to octaves 1, 6, 7, 8 and 9 (Except C6 and B#1) | The following input notes cannot be played as they’re not within Button Bass's keyboard range |