A brief description of notable obstacles you overcame.  
The problem description is quite long and I read it for three times before I fully understood it. During programming, I found it a little hard to separate notes and beats from the tune and detect whether the tune is well-formed or not. Besides, when translating, I also need to separate those notes and judge whether it is playable and whether a beat has more than one note.

A description of the design of your program. You should use pseudocode in this description where it clarifies the presentation.  
  
function isTuneWellFormed;  
 repeatedly:  
 detect the type of the note:  
 if it is a letter,  
 record it and clear other marks;  
 if it is an accidentalSign,  
 if there is a letter immediately before it,  
 record it  
 else  
 tune is not well-formed;  
 if it is a digit,  
 if there is a letter and no digit before it,  
 record it  
 else  
 tune is not well-formed;  
 if it is a slash  
 clear all records;  
 After all the tests is done, if the last character is a ‘/‘,  
 the tune is well-formed  
 else  
 it is not well-formed;  
  
function translateTune;  
 if tune is not well-formed,  
 return 1;  
 else  
 repeatedly: Repeatedly until when? 2 points deducted. More generalization and more focus on important points is needed.  
 check current character:  
 if it is a letter, a sign or a digit,  
 record it;  
 if it is a slash,  
 if current beat is empty,  
 set it as ‘ ‘;  
 else  
 translate the note.  
 if the result is ‘ ‘,  
 it is a badbeat. Assign the number of the badbeat  
 to its corresponding variable and return 2;  
 else   
 add the note to current beat;  
 initialize octave, noteLetter and accidentalSign;  
 if the length of the current beat is larger than 1,  
 add the notes with brackets to the translated tune;  
 else  
 add the note or space to the translated tune;

Nice table. Some more tests and more structure to them, e.g. single beat, would give full points. 2 points deducted.

| tune | instrs | badb | reason | function result | instrs result | badb result |
| --- | --- | --- | --- | --- | --- | --- |
| B#0 | xxx | -999 | Missing ‘/‘. Not well-formed | 1 | xxx | -999 |
| B#0/ | abc | -999 | Not playable | 2 | abc | 0 |
| B#1Cb6C6/Cb2/ | efgijk | 1 | B#1=C2, Cb6=B5, Cb2=B, so the second beat should be bad. | 2 | efgijk | 2 |
| /// | abc | -999 | Beats with no notes. | 0 |  | -999 |
|  | abc | -999 | Empty tune | 0 |  | -999 |
| G3B3DD5//G/A/A3B/C5/B3D5//G//G//CE5//C5/D5/E5/F#5/B3G5//G//G/ | xxx | -999 | General test with a relatively long string to see if the result is totally correct. | 0 | [SFHR] LQ[DW]E[FR] L L [GT] ERT\*[FU] L L | -999 |
| B3#/ | xxx | -999 | Digit and sign’s order is not correct. Not well-formed. | 1 | xxx | -999 |
| #3/ | xxx | -999 | No letter. Not well-formed. | 1 | xxx | -999 |

clear current beat and increase the number of beats;  
 else  
 if the next character is a letter,  
 translate the note;  
 if the result is ‘ ‘,  
 it is a badbeat and return the number of the badbeat;  
 else  
 add the note to current beat;  
 initialize octave, noteLetter and accidentalSign;  
 increase subscript i;  
 assign the translated tune to instructions;  
 return 0;