1 Introduction

This language is used to write sheet music for guitar tabs. Maybe in the future it'll be able to write normal sheet music as well!

2 Design Principles

I want the language to be as simple and intuitive as possible. The syntax is very simple, and a lot of the formatting is taken care of by the evaluator, which means that there are default values for a lot of parameters, making life easier for the user.

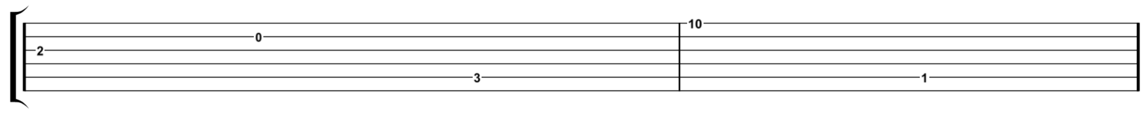
3 Example Programs

Minimal example:

```
-type tab
-time 4-4
-key c

1:
   3e4
   4g
   1g
2:
   5a
   1f
```

Output:



4 Language Concepts

Data types:
There are several data types useful to the user:

Notes - consist of a string, pitch, rhythm, and possible properties
Measure numbers
Specifiers, such as key and time signature

Grammar:

```
<expr>              ::= <option>
                      | <measure>
<option>            ::= <type>
                      | <time>
                      | <key>
<key>               ::= c | cm | c# | c#m| cb | d | dm | db | d#m | e | em | eb | ebm | f |
<time>              ::= <num>-<num>
<num>               ::= x
<type>              ::= tab
<measure>           ::= <note>+
<note>              ::= <simple>
                      | <complex>
                      | <group>
                      | <tuplet>
<simple>            ::= <singlesimple>
                      | <restsimple>
<complex>           ::= <singlecomplex>
                      | <restcomplex>
<singlesimple>      ::= <string><pitch><property>*
<restsimple>        ::= r
<singlecomplex>     ::= <string><pitch><rhythm><property>*
<restcomplex>       ::= r<rhythm>
<group>             ::= (<singlesimple>+)
                      | (<singlesimple>+)<rhythm>
<tuplet>            ::= t<num>o<num> {<simple>+}
<string>            ::= 1 | 2 | 3 | 4 | 5 | 6
<pitch>             ::= | A | ASharp | AFlat | ANat | B | BSharp | BFlat | BNat | C | CSharp
<rhythm>            ::= <rhythmnumber><dot>*
<rhythmnumber       ::= 1 | 2 | 4 | 8 | 16 | 32 | 64
<dot>               ::= .
<property>          ::= /sls | /sle | /stu | /std | /p | /plu | /pld | /gr | /har | /sl | /
```

5 Syntax

A program begins with optional specifiers as to the type of music being written (only tab supported as of now), time signature, and key. They being with a dash, followed by the specifier, and then the actual string
e.g.
-type tab
-key f#
-time 12-8

Measures always being with an int, which is the measure number, followed by a colon
e.g.

4:

Notes start with a string number (1-6), and then a pitch
e.g. 5gb

A rhythm can be specified as: 1 2 4 8 16 32 64
It can be followed by dots as well
e.g.
32..
If no rhythm is given, it defaults to 4

Properties can be specified to change or add qualities. They begin with a / and then the abbreviation
e.g.
/gr means grace note

Notes can also be rests, which are an r followed by an optional rhythm

Examples of valid notes:
5f
4g64../gr/sl
r
r4.

This language doesn't really have associativity issues

6 Semantics

Table 1: Data and Operations

Syntax	**Abstract Syntax**	**Type**	**Meaning**
-type tab	ScoreOption of string * string	string * string	This is an option type, where it begins with a dash, fol
4e32/gr	Complex of int * Pitch * Rhythm * PropertyList	Note	This is a complex note, comprised of all the par

7 Remaining Work