# Extra credit project

## I. Introduction

In this project, you will implement a *chord calculator* based on the grammar in Fig. 1. Given an input song such as the one shown in Fig. 2, the program must first verify whether the input conforms to the grammar. If it does, it should then use the information in Tables I–III to determine the notes that constitute each chord. To evaluate the accuracy of your chord calculator and assign a grade to your project, the program must also display a histogram that counts how many times each note appears in the sequence of chords. The suggested output format is illustrated in Fig. 3.

The input consists of a *song* read from a file. In C, files can be read using the *stdio* library and the function *fgetc*, which reads one character at a time. (For other languages, consult their documentation.) The end of the file is reached when the function returns `EOF`.

A song must contain at least one bar. A bar may begin with a `meter` definition, written as a fraction `num/den`. A bar may contain multiple chords, no chords (indicated by the label "NC"), or repeat the chords of the previous bar (specified with the symbol `%`). Repeated bars must be ignored when computing the total number of notes in the song (last row in Fig. 3). Every bar ends with a vertical line (`|`), as defined in the rule `bar`. After the final bar, a second vertical line marks the end of the song (see rule `song`).

Up to this point, we have described the structure of a song. The following sections explain how to determine the notes that constitute each chord.

```
      input:=  song EOF
       song:=  bar {bar} "|"
        bar:=  [meter] chords "|"
      meter:=  numerator "/" denominator
  numerator:=  "1" | "2" | "3" | ... | "15"
denominator:=  "1" | "2" | "4" | "8" | "16"
     chords:=  "NC" | "%" | chord {chord}
      chord:=  root [description] [bass]
       root:=  note
       note:=  letter [acc]
     letter:=  "A" | "B" | "C" | ... | "G" |
        acc:=  "#" | "b"
description:=  [qual] [qnum] [add] [sus] [omit]
/* qual & sus cannot coexist (one or the other, not both) */
       qual:=  "-" | "+" | "o" | "5" | "1"
       qnum:=  "6" | ["^"] "7" | ["^"] ext
        ext:=  "9" | "11" | "13"
        add:=  alt | "(" alt ")"
        alt:=  [acc] "5" | [acc] ext
        sus:=  "sus2" | "sus4" | "sus24"
       omit:=  "no3" | "no5" | "no35"
       bass:=  "/" note
```

Fig. 1. Chords grammar. The input consists of the chords in a song separated by bar lines "|" plus optional meter information for the bars. Figure 2 shows an example. (This grammar is a subset of the grammar used by Polynizer in its advanced mode: https://www.polynizer.com.)

## II. Root

After the root of a chord is identified, its numeric value or *pitch class* must be computed.[1] (Don't let the term intimidate you: a pitch class is simply the number of the note key in Fig. 4.) The root is the first note you will add to the chord, and it is important to save it in a *variable* because it will be needed in the following steps (everything else is computed in terms of the root).

## III. Quality and suspensions

The quality of a chord determines the next two notes of the chord. To compute them, add (modulo 12) the *intervals* in Table I to the root's pitch class.[2] For example, if the root is F, whose pitch class (key) is 5 (see Fig. 4) and the quality is minor ('-'), you add the notes 8 and 0 to the chord. ($5+3=$ **8** and 5 + 7 = **0** (mod 12)). The *major* chord is special. It has no symbol and may be modified by a *suspension*. A suspension is indicated by "sus2", "sus4", or "sus24", in which case the interval 4 is replaced by 2 (for 'sus2'), 5 (for 'sus4'), or both (for 'sus24'), as shown in Table II.[3]

## IV. Extensions

After the chord quality comes an optional number or *extension*.[4] As with quality and suspensions, an extension adds extra notes to the chord, specified as intervals in Table III. Possible numbers are 6, 7, 9, 11, and 13, representing the corresponding degrees of the scale.[5] Extensions have several characteristics:

[1] See https://en.wikipedia.org/wiki/Pitch_class

[2] Adding something "modulo $n$" means adjusting the result by subtracting or adding $n$ until it lies between 0 and $n-1$. For example, $7+8=3 \pmod{12}$, since subtracting 12 from 15 gives 3, a value between 0 and 11.

[3] The labels ('sus2', 'sus4') refer to scale degrees, while the corresponding intervals in Table II are measured in semitones, as explained earlier for the 3rd and 5th of a chord.

[4] Numbers 1 and 5 were used in Sec. III to represent the unison and the power chord, following musical convention; however, the power chord has no 3rd, so '5' should be considered a symbol rather than a number in the sense used in this section. Unisons consist only of the root, so no extra notes are added.

[5] Strictly speaking, only the 9th, 11th, and 13th are considered *extensions*; for simplicity, the 6th is called extensions here as well.

```
4/4 Db^7 Ab/C C+7 | F-11 Bb9 | Eb9sus4 | Eb13sus4 Eb13 ||
```

Fig. 2. Sample input consisting of the introductory chords of *El día que me quieras*, by Carlos Gardel and Alfredo Le Pera (version by Luis Miguel).

```
     0  1  2  3  4  5  6  7  8  9  A  B
     -  -  -  -  -  -  -  -  -  -  -  -
1.   *  *                 *  *           Db^7
2.   *        *              *           Ab/C
3.   *              *        *     *     C+7
4.   *        *     *        *     *     F-11
5.   *     *        *        *     *     Bb9
6.      *     *     *        *     *     Eb9sus4
7.   *  *     *              *     *     Eb13sus4
8.   *  *     *           *  *     *     Eb13
     -  -  -  -  -  -  -  -  -  -  -  -
     7  4  1  5  1  4  0  1  7  0  6  0
```

Fig. 3. Suggested format for the output. Show the pitch class numbers in the header of a table and mark with a star the notes that make up the chords. Use letters A and B to represent 10 and 11, respectively. (Do not confuse them with note names A and B.) Put the totals at the end of the table. Leave room for two digits in the chord counter (leftmost column) since some songs close to a hundred chords.

TABLE I
CHORD QUALITIES

| | | Intervals (semitones) | |
|---|---|---|---|
| Quality | Symbol | 3rd | 5th |
| Major | none | 4 | 7 |
| Minor | - | 3 | 7 |
| Augmented | + | 4 | 8 |
| Diminished | o | 3 | 6 |
| Power | 5 | | 7 |
| Unison | 1 | | |

Note: The labels "3rd" and "5th" refer to scale degrees, not semitone distances. They do not always match the intervals in the table because two scales are being mixed: degrees and semitones (like Fahrenheit vs. Celsius—different rates of increase and starting points). In a *power* chord, only the 5th is present; this chord is widely used in rock music. The unison consists of a single note: the root.

TABLE II
SUSPENSIONS

| | | Intervals (semitones) | |
|---|---|---|---|
| Suspension | Symbol | Suspension(s) | 5th |
| Second | sus2 | 2 | 7 |
| Fourth | sus4 | 5 | 7 |
| Both | sus24 | 2, 5 | 7 |


Fig. 4. The piano keyboard shows a repeating pattern of 12 keys, numbered 0,1,2,...,11, which can be extended mentally to the right and left. The zigzag pattern is not perfectly regular, since there are no black keys between 4-5 and 11-0′. Letters under the white keys give the note names. Black keys borrow the name of a neighboring white key: append a ♯ (sharp) to the left neighbor or a ♭ (flat) to the right neighbor. For example, key 1 may be called C♯ or D♭. Any note may be altered with a sharp or flat, even if no black key exists, e.g., E♯ = F♭ and B♯ = C♭. Double-sharp (𝄪) and double-flat (𝄫) symbols exist but are not used in this project.

1) The number is optional.
2) Extensions are expressed in degrees, but the intervals in the table are in semitones. For example, a 7th corresponds to 10 semitones.
3) Numbers larger than 7 imply an included 7th.[6] A 7 adds only the 7th, while 9, 11, and 13 implicitly include the 7th as well.
4) If preceded by a caret (^), the 7th is raised one semitone (e.g., 11 semitones instead of 10). The caret is not used with a 6th (i.e., ^6 is invalid).

## V. ADDITIONS

As noted in Sec. IV, the extensions 9th, 11th, and 13th include a 7th, which is usually implied rather than written. In some cases, however, the 7th must be excluded. This is

[6]This is why only 9, 11, and 13 are technically extensions.

TABLE III
EXTENSIONS

| Degree | Abbreviation | Intervals (semitones) | |
|---|---|---|---|
| | | Extension | 7th |
| Sixth | 6 | 9 | |
| Seventh | 7 | | 10 |
| Ninth | 9 | 2 | 10 |
| Eleventh | 11 | 5 | 10 |
| Thirteenth | 13 | 9 | 10 |
| Major seventh | ^7 | | 11 |
| Major ninth | ^9 | 2 | 11 |
| Major eleventh | ^11 | 5 | 11 |
| Major thirteenth | ^13 | 9 | 11 |

indicated by writing the extension in parentheses: '(9)', '(11)', or '(13)'. In such cases, only the intervals 2, 5, and 9 are added, as shown in Table III, while the 10 (i.e., the '7th' column) is omitted.

## VI. ALTERATIONS

Extensions and additions may be altered—raised or lowered—by prepending a sharp (♯) or flat (♭) to the number (e.g., '♯11', '♭9'). Because additions use parentheses, alterations there are unambiguous; with extensions, however, they can be. For example, consider the chord C11 with a raised eleventh. Writing C♯11 would be unclear: is the sharp applied to the root C or to the extension 11? To disambiguate, the chord is written as C7♯11, making the implicit 7th explicit and binding the sharp to the eleventh.

More complex chords follow the same pattern. For instance, C7♭9♯11 is a C major chord with a (minor) seventh, flat ninth, and sharp eleventh. By convention, extensions beyond the first are usually written in ascending order (e.g., C13♭9♯11), though our grammar does not enforce this.

A special case is the flat fifth (♭5). Rather than being added to the chord, it *substitutes* the perfect fifth. For example, C(♭5) is a major triad with a flat fifth. Parentheses are used here to avoid ambiguity, but they are unnecessary in cases like C7♭5♭9.

## VII. OMISSIONS

As said in Sec. III, major and minor chords have a 3rd and a 5th. However, sometimes they are explicitly removed by adding the labels 'no3' (to major chords), 'no5' (to any), or 'no35' (to major chords), the latter for removing both.

## VIII. INVERSIONS

The *bass* (lowest note in a chord) plays a crucial role in music.[7] Notes of a chord can be arranged in any order across instruments or within an instrument's range, but the lowest note is normally the root (see Sec. II), unless explicitly indicated by specifying a bass note after a slash. This is called an *inversion*. For example, if the chord is A♭ but the bass is to play C, it is written as A♭/C. If the bass note does not belong to the chord, it must be added to the list of notes in Fig. 3. This list is circular (no start or end), so the rule that the bass is the leftmost note does not apply here.

[7]Most of the time, the bass is played by an instrument coincidentally called the *bass*, although in some genres it may be a tuba, bassoon, or other instrument.

## IX. EXAMPLES

To illustrate the chord derivation process, we analyze some of the chords in Fig. 3.

**D♭^7** D is key 2 (Fig. 4); applying a flat (♭) reduces it by one, so the root D♭ is key **1**. The chord has no explicit quality ('-', '+', 'o') suspension ('sus2', 'sus4'), or omissions ('no3', 'no5'), so it is major. According to Table I, we add 4 and 7 semitones to the root, giving keys $1+4=$ **5** and $1+7=$ **8**. The ^7 extension adds 11 semitones: $1+11=$ **0** (mod 12).

**A♭/C** A is key 9, so the root A♭ is key **8**. With no quality or suspension specified, this is a major chord. Table I instructs us to add 4 and 7 semitones to the root, yielding keys $8+4=$ **0** (mod 12) and $8+7=$ **3** (mod 12).

**F-11** We start by adding the root F = **5**. The minor quality of the chord implies (by Table I) adding intervals 3 and 7. Hence, we add the notes $5+3=$ **8** and $5+7=$ **0** (mod 12). By Table III, we add 5 and 10 semitones to the root to obtain notes $5+5=$ **10** and $5+10=$ **3** (mod 12).

## X. DELIVERABLES

This is a large project. To assure a good distribution of time, multiple deliverables will be enforced:

10/18. *First* and *follow* sets for the grammar (25 pts).
11/4. Parser (without computing notes that make up the chords) (40 pts).
11/21. Complete chord calculator. (40 pts).

The grades of the parser and the calculator will be based on performance. The parser grade will be proportional to the number of correctly recognized/rejected songs. A total of at least 5 songs will be given to your parser and it will have to recognize which ones are valid according to the grammar. For the case of the calculator, $n$ valid songs will be given to your program and the grade $g$ will be computed based on the $\chi^2$ statistic:[8]

$$g = \frac{40}{12n} \sum_{i=1}^{n} \sum_{j=0}^{11} \left( 1 - \frac{(x_{ij} - x_{ij}^*)^2}{x_{ij}^*} \right),$$

where $x_{ij}$ is the number of times your program found pitch class $j$ ($j = 0, 1, \ldots, 11$) in the $i$th song, $x_{ij}^*$ is the true number, and $n$ is the number of songs used for testing. (The 40 is the number of points the chord calculator is worth from the total).

[8]https://www.investopedia.com/terms/c/chi-square-statistic.asp

For deliverables II and III, a report in PDF is also required. This report must contain should follow the structure of a standard scientific article, including but not limited to the following parts: title, abstract, introduction, methodology, results, conclusions, and bibliography. The details for each section are described next.

*Title:* The title should represent the essence of the work. (Avoid using interrogative titles such as: "How to build a parser for music chords?")

*Abstract:* The abstract summarizes the work in a few lines (usually a single paragraph). It includes the goal of the work, the methodology used to achieve it, the obtained results, and the main conclusions. Each section of the report is summarized here in one or two sentences to form a single paragraph containing between four and eight sentences. Although this section appears first in a report, it is the last to be written (for obvious reasons).

*Introduction:* The introduction explains the goal of the work. It also briefly describes the procedure followed to achieve it. (This section usually contains mostly text and most or all the bibliographic citations).[9]

*Methodology:* The methodology provides a description of the work done to achieve the goal. This is a good place to include the grammar and mention the programming language, compiler, and operating system used. It is not necessary to describe the whole code you wrote to implement the parser, but only its main structure.[10]

*Results:* This section displays the results obtained after running the tests. (Tables and figures are commonly used for this purpose, but for a parser may not be necessary.)

*Conclusions:* This section sets the conclusions drawn from the results.[11] (This section contains mostly text, usually a single paragraph.)

*Bibliography:* This section contains information about the bibliographic sources cited in the text.[12]

It is recommended—but not mandatory—to use the IEEE format for your report.[13] (This project description was made using the IEEE Transactions format).

[9]For this paper, you may only need to cite the course textbook, for example, to indicate that your parser was based on code in that book.

[10]When we say describe the "main structure" we do not mean the *main* function, like in C.

[11]Here, you are expected to indicate if your parser worked, i.e., if it was able to recognize correct songs and reject wrong ones, and to what extent your calculator was able to compute the correct notes.

[12]For this task, you may only need to cite the course textbook.

[13]Specifications for the format and templates for LaTeX and Word are available on the page https://www.ieee.org/publications_standards/publications/authors/author_templates.html.