Exercises:

1. **(Tree Rotation)** In the file `RotateTree.java`, write the methods `rotateLeft` and `rotateRight` to rotate a given node in a BST to the left or the right.

2. **(Music Generation)** A musician is writing a song, and has written a chord sequence, but needs a single note accompaniment (a bass or vocal line, for example). Your task is to generate such an accompaniment.

   The standard Western musical scale is made up of a sequence of notes that correspond to 12 unique tones within an octave; these tones are represented by a repeating sequence of notes denoted by letters from A to G. There is an A note that has twice the frequency of another A note (and therefore has a higher pitch). Furthermore, between each note exists a half-step called a flat or sharp respectively indicated by a b or #. b and # indicate that a note is lowered or raised by a half-step respectively. The spacing between all notes is not even. The following are the 12 unique tones within one octave:

   \[ A, A#/Bb, B, B#/C, C#/Db, D, D#/Eb, E, E#/F, F#/Gb, G, G#/Ab \]

   Notes with slashes can be represented in both forms and are equivalent notes.

   A major scale, given a starting note X, is the sequence of notes (counted in half-steps): X, X+2, X+4, X+5, X+7, X+9, X+11, X+12. Note that X+12 is the same note as X one octave higher. A minor scale consists of the same notes, except the third and sixth notes in the sequence are lowered by one half-step. Chords are sets of notes played simultaneously that are associated with a particular major or minor scale.

   A simple accompaniment can consist of a sequence of notes on a major or minor scale that corresponds with the current chord being played. An accompaniment is uninteresting if it simply plays a scale along with a chord sequence; in general, certain transitions are usually more favorable than others. For example, given a note, the transition from the first note to the fifth note on the scale is often considered to be good.

   Given a chord sequence, transitions and their weights, and the number of notes to be generated to accompany each chord, you are to output the best possible (or one of the best, if multiple exist) accompaniment (sequence of notes) and its total weight, all separated by whitespace; the best such sequence is the one with maximum total weight. Assume the input allows for at least one sequence.

   The first line of input will contain a chord sequence. Minor chords are denoted with a m after the chord name. The second line of input will contain the number of notes you are to generate to accompany each chord. The third line of input will contain the transition values. Each transition value is a pair of integer values X Y where X is the transition distance, \(-11 \leq X \leq 11\), that represents the number of half-steps of the transition, and Y, a positive value that represents the weight of the transition. The line may contain any number of pairs separated by whitespace.

   When generating a sequence for a particular chord, only the transitions that move from the current note to another note on the appropriate scale are considered valid. This means that if you are generating a sequence for C major, and are on the note C, the transition 2 10 can be used to move up to D, but if you are on the note E, the same transition cannot be used to move up to F#. because F# is not on the C major scale.
Place your code in the file `MusicGen.java`

Sample input:

```
C G Am F
4
-7 3 -4 6 -2 7 4 2 5
```

Sample output:

```
E C D E C D E C Ab E C Ab E C D Bb 85
```