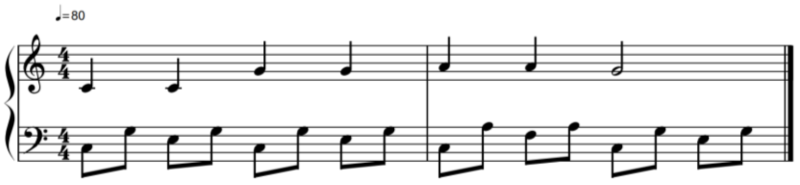
Homework: SuperCollider Basic

This is a homework for you to understand some basic operations of SuperCollider (SC). **<u>Two main problems are included in the homework.</u>**

Problem1. Writing your own sheet music

The goal of this problem is try to code the follow sheet music in the SC code:

Twinkle Twinkle Little Star



This homework is divided into five parts:

(Q1) Try to play the notes of C scale in 80 bpm by **".play(TempoClock(~bpm))"**. Note that ~bpm = 1 is for 60bpm ~bpm = 0.5 is for 30 bpm, etc.

(Q2) Apart from the global tempo speed, we can also control the duration by **"\dur"** in **Pbind**. 1 in **\dur** will be interpreted as the quarter notes (四分音符) and 2 will be interpreted as the 2 half notes (二分音符), etc. Let's try to play the tones in [Q,Q,H,Q,Q,H,Q,Q] pattern, where Q means quarter notes and H means half notes.

(Q3) Last, we can also try to control the pitch of each note. In this case, we use **"\degree"** to describe the pitch of note. Let's try to write a descending C major scale.

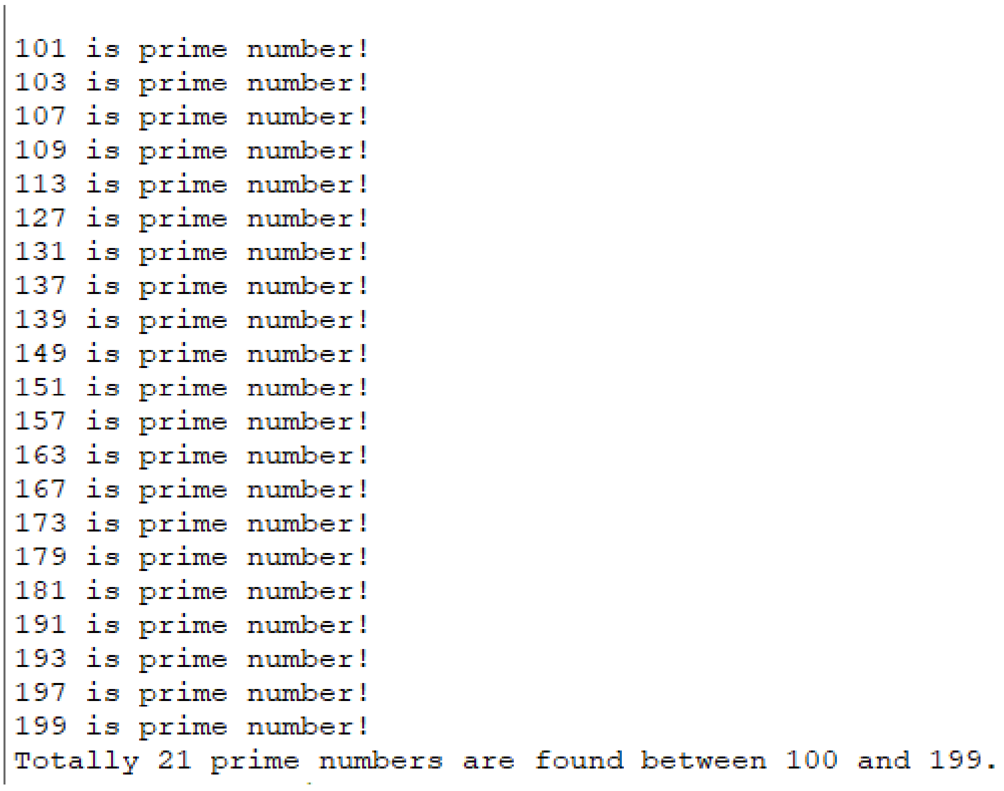
(Q4) & (Q5) Now, we can start to code Twinkle Twinkle Little Star for both the two hands part. You are encouraged to utilize "repeat" argument in Pseq to make you code simpler.

Problem2. More about SC languages

There are two parts in “Problem2.scd”. In Part1, you can see four subparts about “Array operations”, “Conditionals” and “Functions”. Please run each line of codes or each unit and you will see results on Post window. **<u>You don’t need to answer any question in this part, but make sure you understand what they do.</u>** To help you get familiar with them, it’s recommended to adjust arguments or parameters.

In Part2, you are asked to design a program to find all “Prime numbers” between 500 and 999. When you finish it, you should print:
(1) Every Prime number between 500 and 999
(2) How many Prime numbers did you find?

For example:
If the range of Prime numbers is 100~199, the result on post window will be like the figure below.



```
101 is prime number!
103 is prime number!
107 is prime number!
109 is prime number!
113 is prime number!
127 is prime number!
131 is prime number!
137 is prime number!
139 is prime number!
149 is prime number!
151 is prime number!
157 is prime number!
163 is prime number!
167 is prime number!
173 is prime number!
179 is prime number!
181 is prime number!
191 is prime number!
193 is prime number!
197 is prime number!
199 is prime number!
Totally 21 prime numbers are found between 100 and 199.
```

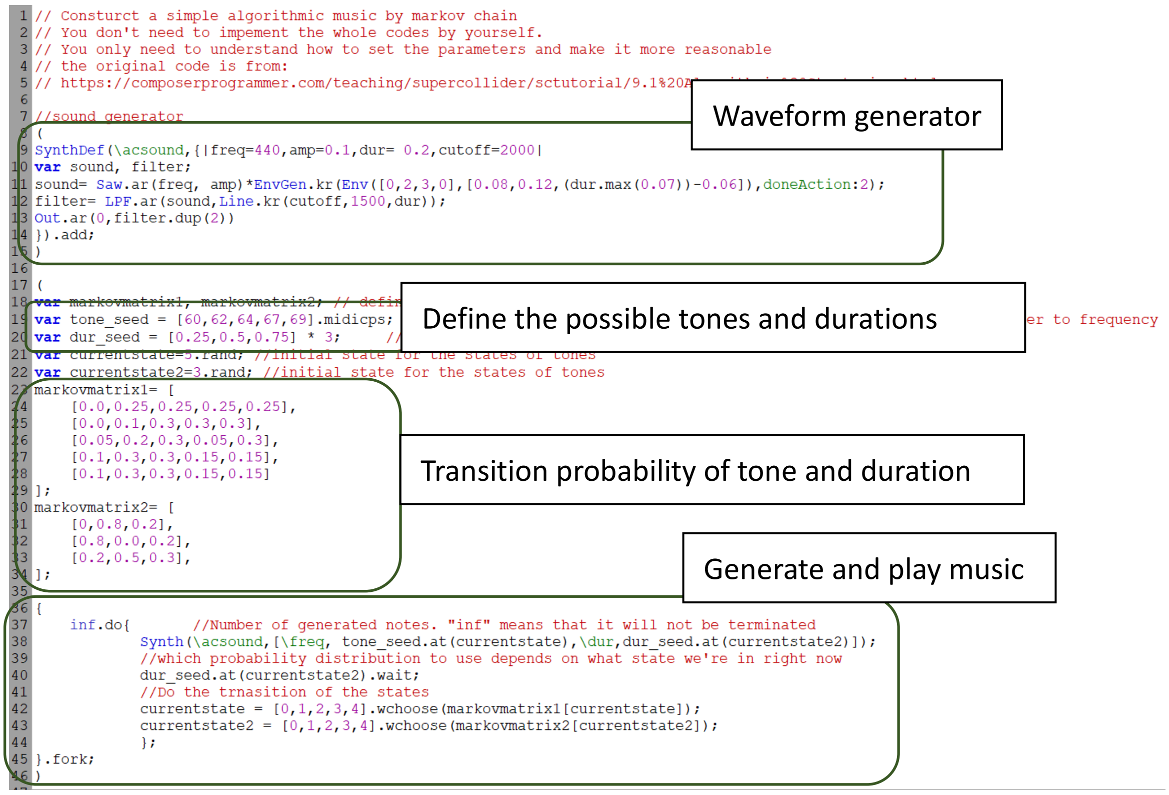
Please write your program in Part2 of Problem2.scd. If you finish this Problem, save it as **Problem2_yourID_yourName.scd**.

*You may write your code based on what you've learned in Part1, but it's okay for you to use any method in Supercollider to solve this problem.
*If Part1 cannot help you solve Part2 perfectly, try to search other SC related websites.

Bonus. Practice to trace codes and modified it

Many of excellent codes are shared on online. To understand the codes from others and modify it can help you to finish your work more quickly.

In this problem, we write a simple program to do algorithmic composition by Markov chain. The code is also modified from the code written by another. Your mission is making the generated music more beautiful/interesting/euphonic…, etc. The answer is not unique and without standards. For example, you can create a new Markov chain and make the two tracks play harmonically in tune.


Waveform generator
Define the possible tones and durations
Transition probability of tone and duration
Generate and play music

```
// Consturct a simple algorithmic music by markov chain
// You don't need to impement the whole codes by yourself.
// You only need to understand how to set the parameters and make it more reasonable
// the original code is from:
// https://composerprogrammer.com/teaching/supercollider/sctutorial/9.1%20A

//sound generator
(
SynthDef(\acsound,{|freq=440,amp=0.1,dur= 0.2,cutoff=2000|
var sound, filter;
sound= Saw.ar(freq, amp)*EnvGen.kr(Env([0,2,3,0],[0.08,0.12,(dur.max(0.07))-0.06]),doneAction:2);
filter= LPF.ar(sound,Line.kr(cutoff,1500,dur));
Out.ar(0,filter.dup(2))
}).add;
)

(
var markovmatrix1, markovmatrix2; // defi
var tone_seed = [60,62,64,67,69].midicps;                                    er to frequency
var dur_seed = [0.25,0.5,0.75] * 3;      //
var currentstate=5.rand; //initial state for the states of tones
var currentstate2=3.rand; //initial state for the states of tones
markovmatrix1= [
    [0.0,0.25,0.25,0.25,0.25],
    [0.0,0.1,0.3,0.3,0.3],
    [0.05,0.2,0.3,0.05,0.3],
    [0.1,0.3,0.3,0.15,0.15],
    [0.1,0.3,0.3,0.15,0.15]
];
markovmatrix2= [
    [0,0.8,0.2],
    [0.8,0.0,0.2],
    [0.2,0.5,0.3],
];

{
    inf.do{       //Number of generated notes. "inf" means that it will not be terminated
            Synth(\acsound,[\freq, tone_seed.at(currentstate),\dur,dur_seed.at(currentstate2)]);
            //which probability distribution to use depends on what state we're in right now
            dur_seed.at(currentstate2).wait;
            //Do the trnasition of the states
            currentstate = [0,1,2,3,4].wchoose(markovmatrix1[currentstate]);
            currentstate2 = [0,1,2,3,4].wchoose(markovmatrix2[currentstate2]);
            };
}.fork;
)
```

*The original code is from here:
https://composerprogrammer.com/teaching/supercollider/sctutorial/9.1%20Algorithmic%20Strategies.html