학번	
이름	

1. [5점] Hz 단위의 실수로 표현하는 주파수(frequency)에 대한 다음 설명 중에서 맞는 것을 모두 고르자.

   (1) 음파의 주기 패턴의 폭이 좁을 수록 주파수가 크다.
   (2) 음파의 주기 패턴의 폭이 넓을 수록 주파수가 크다.
   (3) 음파의 상하 높낮이의 길이가 길수록 주파수가 크다.
   (4) 음파의 상하 높낮이의 길이가 짧을수록 주파수가 크다.
   (5) 주파수가 크면 고음이 나고, 주파수가 작으면 저음이 난다.
   (6) 주파수가 크면 저음이 나고, 주파수가 작으면 고음이 난다.

2. [5점] 주파수가 440Hz인 음파의 한 주기(cycle) 기간은 몇 초 인지 소수점 5째 자리 까지 정확도로 계산하자.

1 / 440 = 0.00227(초)

3. [5점] 다음 음파 중에서 하나만 있으면 다른 음파를 모두 만들 수 있다. 그 음파는 무엇인가?

   (1) 사인파(sine wave)
   (2) 삼각파(triangle wave)
   (3) 톱니파(sawtooth wave)
   (4) 사각파(square wave)

4. [5점] 다음 프로그램을 실행하면 콘솔창에 어떤 값이 프린트 되는지 예측하자.

```
SqrOsc s => dac;
<<< s.freq(), s.gain() >>>;
```

220.0 1.0

5. [5점] 다음 프로그램을 실행하면 몇 초 동안 소리가 날지 예측하자.

```
SinOsc s => dac;
5*(4/5)::second => now;
```

0초

6. [5점] 아래 두 프로그램을 실행하여 나는 소리는 같은가? 같으면 왜 같고, 다르면 왜 다른지 간단히 설명하자.

프로그램 1

```
SinOsc s => dac;
for (0 => int i; i < 8; i++) {
    Math.random2(48,60) => float note;
    Std.mtof(note) => s.freq;
    0.5::second => now;
}
```

프로그램 2

```
SinOsc s => dac;
for (0 => int i; i < 8; i++) {
    Math.random2f(48,60) => float note;
    Std.mtof(note) => s.freq;
    0.5::second => now;
}
```

다르다. 프로그램1은 정수만 무작위로 생성하는데 반해, 프로그램2는 실수를 무작위로 생성하므로 생성하는 수의 범위가 더 넓다.

7. [5점] 아래 두 프로그램의 실행 의미 차이점을 간단히 설명하자.

프로그램 1

```
SinOsc s => dac;
for (0 => int i; i < 8; i++) {
    Math.random2(48,60) => int note;
    Std.mtof(note) => s.freq;
    0.5::second => now;
}
```

프로그램 2

```
SinOsc s => dac;
Math.srandom(58);
for (0 => int i; i < 8; i++) {
    Math.random2(48,60) => int note;
    Std.mtof(note) => s.freq;
    0.5::second => now;
}
```

프로그램1은 실행할 때마다 다른 소리를 내지만, 프로그램2는 실행하면 항상 같은 소리를 낸다.

8. [5점] 다음 프로그램을 실행하면 콘솔 창에 어떤 값을 프린트할지 예측하자.

```
[48, 50, 52, 53, 55, 57, 59, 60] @=> int notes[];
12 +=> notes[3];
notes.size(5);
notes << notes[2] << notes[4];
notes.size(8);
notes.popBack();
for (0 => int i; i < notes.size(); i++)
    <<< notes[i] >>>;
```

48
50
52
65
55
52
55

9. [5점] 스테레오(채널 2개)의 아날로그 소리를 디지털로 바꾸면서 초당 44,100개의 샘플로 샘플 당 16비트(= 2 바이트)를 쓰는 경우, 1 시간 짜리 음악을 저장하기 위한 용량을 메가바이트 단위로 계산하자. (편의상, 1 메가바이트 = 1,000 킬로바이트, 1 킬로바이트 = 1,000 바이트로 계산하고, 소수점 아래는 버려도 좋다. 정확하게 계산하려면 1,000 대신, 1,024를 써야 하지만.. 종이 시험이니까..)

2 x 44,100 x 2 x 60 x 60 / 1,000 / 1,000
= 635 Megabyte

10. [5점] 다음 프로그램을 실행하니 소리가 나지 않는다. 가능한 원인을 두 가지 찾아서 기술하자.

```
SndBuf sample => dac;
me.dir() + "/audio/snare_01.wav" => string filename;
filename => sample.read;
sample.samples() => sample.pos;
second => now;
```

0 => sample.pos;
"/audio/snare_01.wav"에 파일이 없거나 파일 경로가 틀렸다.

11. [5점] 아래 두 프로그램이 내는 소리의 차이점을 간단히 기술하자.

프로그램 1

```
SndBuf stereo_sample => dac;
me.dir() + "/audio/stereo_fx_01.wav" => stereo_sample.read;
stereo_sample.length() => now;
```

프로그램 2

```
SndBuf2 stereo_sample => dac;
me.dir() + "/audio/stereo_fx_01.wav" => stereo_sample.read;
stereo_sample.length() => now;
```

SndBuf2는 스테레오 두 채널로 분리되어 있는 샘플을 읽어서 각 채널을 dac.left와 dac.right에 각각 연결하여 내보낼 수 있다. SndBuf는 그렇지 못하기 때문에 dac.left와 dac.right에 연결하더라도 두 채널의 소리가 합쳐져서 양쪽 스피커에 같은 소리가 들릴 뿐이다.

12. [5점] 다음 진동기 UGen 중에서 PulseOsc와 가장 비슷한 UGen을 하나 고르자.
    (1) SinOsc
    (2) TriOsc
    (3) SawOsc
    (4) SqrOsc

13. [5점] dac, adc, blackhole Ugen에 대한 설명 중에서 틀린 것을 하나 고르자.
    (1) 따로 선언해 두지 않아도 항상 존재하는 빌트인(built-in) UGen이다.
    (2) adc => dac; 명령으로 한번 연결해두면, adc =< dac 명령으로 끊지 않는 한, 버철머신이 가동하는 동안 계속 연결이 되어 있다.
    (3) dac는 마이크로 들어오는 아날로그 소리를 디지털 데이터로 변환해주는 역할을 하고, adc는 디지털 소리 데이터를 아날로그 소리로 변환하여 스피커로 내보내는 역할을 한다.
    (4) blackhole은 처리한 소리 테이터를 스피커로 내보내고 싶지 않을 때 유용하게 사용하는 UGen 이다.

14. [5점] Envelope UGen의 역할을 바르게 설명한 것이 아닌 것을 하나 고르자.
    (1) 소리를 부드럽게 시작하고 부드럽게 끝나게 해준다.
    (2) ADSR은 Envelope UGen의 일종이다.
    (3) 소리 크기 목표치를 지정할 수 있다.
    (4) 소리가 멀리 퍼져나가지 못하게 막아준다.

15. [5점] 주파수 변조(Frequency Modulation)를 통한 소리 합성에 대해서 틀리게 설명한 것을 하나 고르자.
    (1) 주파수가 다른 소리를 합하면 완연히 다른 소리가 나는데 착안한 소리 합성 방법이다.
    (2) Sine 파형 만으로 주파수 변조하여 다양한 악기 소리를 합성할 수 있다.
    (3) 변조하는 대상이 되는 두 개의 소리의 주파수가 정확히 정수 비율인 경우에만 화음이 맞는 소리가 합성된다.
    (4) ChucK 프로그램으로는 새로운 소리 합성이 불가능하므로, 미리 전문가가 주파수 변조하여 만들어 놓은 라이브러리 FM UGen을 사용하는 수밖에 없다.

16. [5점] StkInstrument의 앞부분 STK는 무엇의 약자인가?
    (1) Sound Tool Kit
    (2) Synthesis Tool Kit
    (3) Standard Tool Kit
    (4) Spectrum Tool Kit

17. [10점] 아래 프로그램은 가운데 음(MIDI 60)에서 시작하여 반음 올리거나, 내리거나, 그대로 두거나 중에 무작위로 하나를 선택하여 0.3초씩 소리를 무한대로 내는 프로그램이다. 위아래로 음의 상한과 하한을 각각 72와 48로 두었다. 이 프로그램을 다음 두 요구사항을 만족하도록 수정하자.

요구사항 1 - 같은 음이 연속해서 소리나지 않게 한다. 즉, 다음 음은 반드시 올리거나, 내리거나, 둘 중에 하나를 무작위로 선택한다.
요구사항 2 - 음이 상한 또는 하한에 도달했을 때도, 같은 음이 연속해서 나지 않아야 한다.

```
SinOsc s => dac;
60 => int note;
while (true) {
    Std.mtof(note) => s.freq;
    0.3::second => now;
    Math.random2(-1,1) +=> note;
    if (note < 48) 48 => note;
    if (note > 72) 72 => note;
}
```

답:

```
SinOsc s => dac;
60 => int note;
int tossed;
while (true) {
    Std.mtof(note) => s.freq;
    0.3::second => now;
    Math.random2(0,1) => tossed;
    if (tossed == 0)
        -1 => tossed;
    tossed +=> note;
    if (note < 48 || note > 72)
        tossed * 2 -=> note;
}
```

18. [10점] 다음 프로그램을 읽고 아래 물음에 답하자.

```
SqrOsc s => dac;

0.5::second => dur BEAT;
BEAT => dur QN;
QN / 2 => dur HQN;
0.5 => float VOLUME;

70 => int CENTERNOTE;
int note;
for (0 => int i; i < 8; i++) {
    Math.random2(CENTERNOTE-12, CENTERNOTE+12) => note;
    makeSound(s, Std.mtof(note), VOLUME, HQN);
    makeSound(s, Std.mtof(note+2), VOLUME, HQN);
    makeSound(s, Std.mtof(note+3), VOLUME, HQN);
    makeSound(s, Std.mtof(note-1), VOLUME, HQN);
    makeSound(s, Std.mtof(note), VOLUME, HQN);
    makeSound(s, Std.mtof(note-4), VOLUME, HQN);
    if (Math.random2(1,2) == 1) {
        makeSound(s, Std.mtof(note-5), VOLUME/2, QN);
        makeSound(s, Std.mtof(note-5), VOLUME/2, QN);
        makeSound(s, Std.mtof(note-5), VOLUME/2, QN);
    }
    else
        BEAT * 3 => now;
}

fun void makeSound(Osc osc, float pit, float vol, dur len) {
    pit => osc.freq;
    vol => osc.gain;
    len * 4 / 5 => now;
    0 => osc.gain;
```

```
        len / 5 => now;
    }
```

다음 프로그램은 SqrOsc의 소리를 Envelope를 통과시켜 음을 부드럽게 하여 실제 악기를 부는 소리와 가깝게 한 사례이다. 이 프로그램을 참고하여 위 프로그램에 Envelope를 활용하여 소리를 개선해보자. (수정해야 할 부분을 긋고 여백을 활용하여 새 코드를 적어 넣으세요.)

```
    SqrOsc clarinet => Envelope env => dac;
    [48,50,52,53,55,57,59,60] @=> int scale[];
    for (0 => int i; i < scale.size(); i++) {
        Std.mtof(scale[i]) => clarinet.freq;
        1 => env.keyOn;
        0.25::second => now;
        1 => env.keyOff;
        0.25 :: second => now;
    }
```

답: 수정할 부분

```
SqrOsc s => Envelope env => dac;

fun void makeSound(Osc osc, float pit, float vol, dur len) {
    pit => osc.freq;
    vol => osc.gain;
    1 => env.keyOn;
    len * 4 / 5 => now;
    1 => env.keyOff;
    len / 5 => now;
}
```