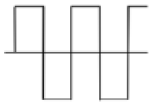
학번	
이름	

1. [5점] 물체가 진동하여 생기는 음파의 진폭(amplitude)에 대한 설명 중에서 맞는 것을 모두 고르자.

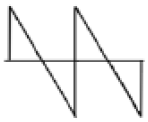
   (1) 음파 진동 주기 패턴의 가로 축 길이를 진폭이라고 한다.
   (2) 음파 진동 주기 패턴의 세로 축 길이를 진폭이라고 한다.
   (3) 진폭이 길수록 고음이 난다.
   (4) 진폭이 길수록 저음이 난다.
   (5) 진폭이 길수록 큰 소리가 난다.
   (6) 진폭이 길수록 작은 소리가 난다.

(정답) (2), (5)

2. [5점] 다음 파형 별로 이름을 오른쪽에서 고르자.

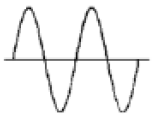


(1) sine wave



(2) triangle wave

(3) sawtooth wave



(4) square wave

(정답) (4) (3) (2) (1)

3. [5점] 음파의 진동 주기가 0.01초 이면, 주파수는 몇 Hz 일까?

(정답) 100 Hz

4. [5점] 다음 프로그램을 실행하면 몇 초 동안 소리가 날지 예측하자.

```
SinOsc s => dac;
(5+1/2)::second => now;
```

(정답) 5초

5. [5점] 다음 프로그램의 실행 결과를 예측하자.

```
SinOsc s => dac;
s.freq() * 2 => s.freq;
s.gain() / 2 => s.gain;
<<< s.freq(), s.gain() >>>;
```

(정답) 440.0 0.5

6. [5점] 다음 중에서 허용하는 지정문을 모두 고르자.

```
(1) 220 => int hz;
(2) 220 => float hz;
(3) 220.0 => int hz;
(4) 220.0 => float hz;
(5) 220.0 $ int => int hz;
(6) 220.0 $ int => float hz;
```

(정답) 1, 2, 4, 5, 6

7. [5점] 아래 두 프로그램에서 n 변수가 갖게 되는 값의 차이점을 간단히 설명하자.

프로그램 1
```
Math.random2(0,1) => float n;
```
프로그램 2
```
Math.random2f(0,1) => float n;
```

(정답) 프로그램1에서는 0.0 또는 1.0 둘 중 하나를 무작위로 갖는데 비해, 프로그램2에서는 0.0과 1.0 범위의 모든 실수 중 하나를 무작위로 갖는다.

8. [5점] 다음 프로그램의 실행 결과를 예측하자.

```
int notes[];
for (0 => int i; i < notes.size(); i++)
    <<< notes[i], "" >>>;
```

(정답) 배열이 존재하지 않으므로 배열 접근이 불가하다는 실행 오류가 발생한다.

9. [5점] 다음 프로그램의 실행 결과를 예측하자.

```
int notes[4];
for (0 => int i; i <= notes.size(); i++)
    <<< notes[i], "" >>>;
```

(정답) 0을 4번 프린트한 후, ArrayOutOfBounds 실행 오류가 발생한다.
0
0
0
0

10. [5점] 다음 프로그램을 실행하면 콘솔 창에 어떤 값을 프린트할지 예측하자.

```
[48, 50, 52, 53, 55, 57, 59, 60] @=> int notes[];
Std.mtof(notes[4]) => float freq;
notes[5] => int note1;
notes[7] => int note2;
notes.size(4);
notes << note1 << note2;
if (note1 < note2)
    notes << note1;
else
    notes << note2;
notes.size(8);
Std.ftom(freq) $ int  => notes[7];
<<< notes[4], notes[5], notes[6], notes[7] >>>;
```

(정답) 57 60 57 55

11. [5점] Osc 진동기와 음정, 소리크기를 인수로 받아서 그 진동기의 주파수와 소리크기를 설정하는 함수를 다음과 같이 작성하였다. 아래 중에서 맞는 것을 **모두** 고르자.

```
fun void setNote(Osc osc, float freq, float volume) {
    freq => osc.freq;
    volume => osc.gain;
}

fun void setNote(Osc osc, int note, float volume) {
    Std.mtof(note) => osc.freq;
    volume => osc.gain;
}
```

(1) SqrOsc clarinet; setNote(clarinet, 330.0, 1.0); 와 같이 호출하면 오류가 발생하지 않고 의도한 대로 잘 작동한다.

(2) SawOsc violin; setNote(violin, 60, 1); 와 같이 호출하면 오류가 발생하지 않고 의도한 대로 잘 작동한다.

(3) 위 두 함수는 함수 이름이 같지만 파라미터의 타입이 다르므로 공존이 가능하다.

(4) 위 두 함수는 함수 이름이 같기 때문에 둘 중 하나는 함수 이름을 다르게 수정해야 한다.

(정답) (1), (3)

12. [5점] 다음 프로그램에서 첫째 for 루프와 둘째 for 루프가 내는 소리의 차이를 글로 설명하자.

```
SndBuf sample => dac;
me.dir() + "audio/hihat_01.wav" => sample.read;

for (0 => int i; i < 3; i++) {
    0 => sample.pos;
    sample.length() => now;
}

for (0 => int i; i < 3; i++) {
    -1.0 => sample.rate;
    sample.samples() => sample.pos;
    sample.length() => now;
}
```

(정답) 첫째 루프는 hihat_01.wav 파일을 앞에서부터 정방향으로 읽어 세번 연속 세번 내는 반면, 둘째 루프는 뒤에서부터 역방향으로 세번 연속 소리를 낸다.

13. [5점] 마이크와 스피커를 `adc => dac;` 과 같이 직접 연결하지 않고, `adc => Gain g => dac;` 와 같이 중간에 `Gain` UGen을 둠으로써 얻을 수 있는 장점을 기술하자.

(정답) 직접 연결해두면, 프로그램 종료 후에도 adc와 dac의 연결 상태가 지속된다. 반면 중간에 Gain 을 두어 연결하면, 프로그램 종료와동시에 adc와 dac의 연결이 저절로 끊어지므로 명시적으로 끊을 필요가 없어서 편리하다.

14. [5점] 다음 ADSR 네 구간 중에서 시간 단위가 아닌 것은?

(1) Attack
(2) Decay
(3) Sustain
(4) Release

(정답) (3)

15. [5점] 다음 두 프로그램이 내는 소리가 어떻게 다른지 설명하자.

프로그램 1

```
Impulse imp => Delay str => dac;
str => str;
441.0::samp => str.delay;
1.0 => imp.next;
2::second => now;
```

프로그램 2

```
Impulse imp => Delay str => dac;
str => str;
441.0::samp => str.delay;
0.98 => str.gain;
1.0 => imp.next;
2::second => now;
```

(정답) 프로그램1은 펄스 소리가 0.01초에 한번씩 2초동안 200번 같은 볼륨으로 나는 반면, 프로그램2는 펄스 소리가 같은 간격으로 반복되기는 하지만 볼륨이 점점 작아지면서 곧 사라진다.

16. [5점] 다음 중에서 소리 파일을 담을 수 있는 UGen을 모두 고르자.

(1)Gain
(2)Pan2
(3)SndBuf
(4)SndBuf2

(정답) (3), (4)

17. [10점] 이 프로그램을 먼저 이해하고 아래 요구사항 대로 코드를 수정하자.

```
SndBuf kick => Gain master => dac;
SndBuf snare => master;
SndBuf hihat => master;
SawOsc voice => master;

me.dir()+"/audio/kick_01.wav" => kick.read;
me.dir()+"/audio/snare_01.wav" => snare.read;
me.dir()+"/audio/hihat_01.wav" => hihat.read;

0.3 => hihat.gain;
0.15 :: second => dur tempo;

[1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0] @=> int kickHits[];
[0,0,1,0,0,0,1,0,0,0,0,0,1,1,1,1] @=> int snareHits[];
[0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1] @=> int hihatHits[];

for (0 => int i; i < 8; i++)
    for (0 => int beat; beat < kickHits.size(); beat++) {
        if (kickHits[beat])
            0 => kick.pos;
        if (snareHits[beat])
            0 => snare.pos;
        if (hihatHits[beat])
            0 => hihat.pos;
        Math.random2(0,2) => int play;
        Math.random2(60,72) => int note;
        if (play == 0) {
            Std.mtof(note) => voice.freq;
            0.3 => voice.gain;
        }
        else
            0.0 => voice.gain;
        tempo => now;
    }
```

위 프로그램을 다음 두 함수를 활용하도록 for 루프의 몸체 코드블록을 수정하자. 두 함수를 모두 호출하여야 하며, 수정한 프로그램은 수정 전과 똑같이 연주되어야 한다. (수정해야 할 부분을 긋고 여백을 활용하여 새 코드를 적어 넣는다.)

```
fun void playDrum(SndBuf drum, int hits[], int beat) {
    if (hits[beat])
        0 => drum.pos;
}

fun void leadVoice(Osc sound, int distance, float volume, int low, int high) {
    Math.random2(0,distance-1) => int play;
    Math.random2(low,high) => int note;
    if (play == 0) {
        Std.mtof(note) => sound.freq;
        volume => sound.gain;
    }
    else
        0.0 => sound.gain;
}
```

(정답) for 루프만 다음과 같이 변경

```
    for (0 => int i; i < 8; i++)
        for (0 => int beat; beat < kickHits.size(); beat++) {
            playDrum(kick, kickHits, beat);
            playDrum(snare, snareHits, beat);
            playDrum(hihat, hihatHits, beat);
            leadVoice(voice, 3, 0.3, 60, 72);
            tempo => now;
        }
```

18. [10점] 다음 프로그램을 읽고 아래 물음에 답하자.

```
SawOsc 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 SawOsc의 소리를 ADSR를 통과시켜 음을 부드럽게 하여 실제 악기를 부는 소리와 가깝게 한 사례이다. 이 프로그램을 참고하여 위 프로그램에 ADSR를 활용하여 소리를 개선해보자. (수정해야 할 부분을 긋고 여백을 활용하여 새 코드를 적어 넣는다.)

```
SawOsc s => ADSR env => dac;
[48,50,52,53,55,57,59,60] @=> int scale[];
env.set(0.1::second, 0.05::second, 0.5, 0.1::second);

for (0 => int i; i < scale.size(); i++) {
    Std.mtof(scale[i]) => s.freq;
    1 => env.keyOn;
    0.4::second => now;
    1 => env.keyOff;
    0.1 :: second => now;
}
```

(정답) 수정할 부분

```
SawOsc s => ADSR env => dac;
env.set(0.1::second, 0.05::second, 0.5, 0.1::second);

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;
}
```