학번	
이름	

1. [10점] ChucK 프로그램의 쉬레드(shred)에 대한 다음 설명 중에서 틀린 것을 모두 고르시오.
   (1) spork ~ f(); 와 같이 기술하면 함수 f()의 호출을 실행하는 자식 쉬레드가 생긴다.
   (2) Machine.add(path); 와 같이 기술하면 path로 지정한 파일의 코드를 실행하는 새로운 쉬레드가 생긴다.
   (3) 만들 수 있는 자식 쉬레드의 수에는 제한이 없다.
   (4) 부모 쉬레드가 없어지면, 자식을 포함한 자손 쉬레드도 모두 없어진다.
   (5) 자식 쉬레드끼리 시간에 맞추어 동기화할 수 있으며, 동기화 시기는 프로그램으로 결정할 수 있다.
   (6) 자식 쉬레드가 하나라도 없어지면 부모 쉬레드도 사라진다.

(정답) (6)

2. [10점] class 키워드 앞에 public 이라고 명시하는 이유를 하나만 쓰시오

(정답) 명시하지 않으면 다른 파일에서 그 클래스를 접근할 수 없어 해당 객체를 생성할 수 없기 때문이다.

3. [10점] 필드 변수의 앞에 static 으로 언급한 변수와 언급하지 않은 변수의 다른점을 설명하시오.

(정답) static 이 아닌 필드 변수는 객체 소속으로, 설정한 값은 생성한 객체가 살아있는 동안만 유효하고 실행이 끝나면 객체와 함께 사라지지만, static 필드 변수는 클래스 소속으로, 객체의 생사 여부와 상관없이 영구히 존재한다. 따라서 static 필드 변수 값은 버철 머신 전체에서 공유된다.

4. [10점] 다음 프로그램을 실행하면 매초 어떤 소리가 날지 예측하여 말로 기술하시오.

```
class MyClarinet extends Clarinet {

    fun void noteOn(int note, float volume) {
        Std.mtof(note) => this.freq;
        volume => this.noteOn;
    }
}

MyClarinet clarinet => dac;

clarinet.noteOn(1.0);
second => now;
clarinet.noteOff(1.0);
second => now;
clarinet.noteOn(72, 1);
second => now;
```

(정답) 첫 1초는 기본값 (주파수 220.0, 볼륨 1.0)으로 클라리넷 소리가 나고, 다음 1초 동안은 소리가 나지 않고, 다음 1초 동안은 MIDI 72 음으로 클라리넷 소리가 난다.

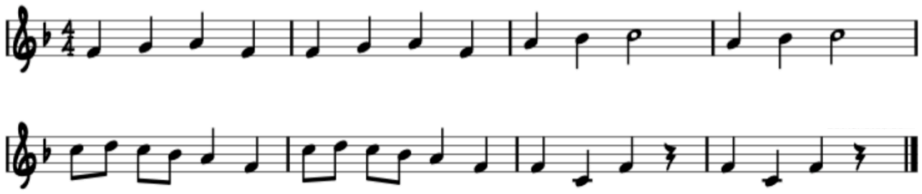
5. [10점] Event e 객체에 신호를 보내는 두 가지 방법 e..signal()과 e.broadcast()의 다른 점을 설명하시오.

(정답) e.signal()은 Event e 발생을 기다리는 객체의 생성 순으로 하나에게 만 신호를 보내지만, e.broadcast()는 e의 발생을 기다리는 모두에게 한꺼번에 신호를 보낸다.

6. [10점] 다음 코드는 어떤 용도로 사용하는지 설명하시오.

```
me.yield();
```

(정답) 부모 쉬레드가 이 코드를 실행하면, 자신이 시간을 보내지 않고도 모든 자식이 즉시 실행을 하도록 풀어준다.



아래 두 쪽에 걸쳐 있는 프로그램을 실행하면 위 악보를 한 번 연주한다. 이 프로그램을 이해한 다음 아래 물음에 답하시오.

```
class BPM { // Beats Per Minute

    static dur qn, en, hn;

    fun void tempo(float beat) { // beat in BPM
        60.0 / beat => float spb; // seconds per beat
        spb::second => qn; // 1/4
        qn / 2.0 => en; // 1/8
        qn * 2.0 => hn; // 1/2
    }
}

class Play {

    fun void play(StkInstrument instrument, string notes[], dur durs[]) {
        for (0 => int i; i < notes.size(); i++)
            playnote(instrument, notes[i], durs[i]);
    }

    fun void playnote(StkInstrument instrument, string note, dur duration) {
        if (note != "--") {
            Std.mtof(s2n(note)) => instrument.freq;
            1 => instrument.noteOn;
        }
        duration => now;
        1 => instrument.noteOff;
    }

    fun int s2n(string name) {
        [21,23,12,14,16,17,19] @=> int notes[]; // A0,B0,C0,D0,E0,F0,G0
        name.charAt(0) - 65 => int base; // A=0,B=1,C=2,D=3,E=4,F=5,G=7
        notes[base] => int note;
        if (0 <= base && base <= 6) {
            if (name.charAt(1) == '#' || name.charAt(1) == 's') // sharp
                notes[base] + 1 => note;
            if (name.charAt(1) == 'b' || name.charAt(1) == 'f') // flat
                notes[base] - 1 => note;
        }
        else
            <<< "Illegal Note Name!" >>>;
        name.charAt(name.length()-1) - 48 => int oct; // 0, 1, 2, ..., 9
        if (0 <= oct && oct <= 9) {
            12 * oct +=> note;
            return note;
        }
        else
            <<< "Illegal Octave!" >>>;
    }
}
```

```
BPM t;
t.tempo(150);
t.qn => dur qn;
t.hn => dur hn;
t.en => dur en;
Play p;

[
"F4","G4","A4","F4",             "F4","G4","A4","F4",
"A4","Bb4","C5",                 "A4","Bb4","C5",
"C5","D5","C5","Bb4","A4","F4", "C5","D5","C5","Bb4","A4","F4",
"F4","C4","F4","--",             "F4","C4","F4","--"
] @=> string melody[];

[
 qn,  qn,  qn,  qn,              qn,  qn,  qn,  qn,
 qn,  qn,  hn,                   qn,  qn,  hn,
 en,  en,  en,  en,   qn,  qn,   en,  en,  en,  en,   qn,  qn,
 qn,  qn,  qn,  qn,              qn,  qn,  qn,  qn
] @=> dur durs[];

Rhodey piano => dac;
p.play(piano, melody, durs);
t.qn * 32 => now;
```

7. [20점] 4개의 피아노가 위 악보를 돌림으로 연주하도록 프로그램을 수정하자. 각 피아노는 2 마디의 시차를 두고 연주를 시작한다. spork를 활용하여 멜로디를 연주하는 쉬레드를 4개 만들어야 한다. 제거할 부분을 위 코드에 표시하고, 아래 빈칸에 추가할 코드를 작성한다.

8. [20점] 이번에는 똑같은 돌림노래를 spork 대신 Machine.add 를 활용하여 재작성해보자. BPM.ck, Play.ck, melody.ck, starter.ck는 모두 아래와 같이 완성되어 있고, score.ck만 완성하면 된다. 5개 파일은 모두 같은 폴더에 들어있다고 가정한다.

```
// BPM.ck
public class BPM { // Beats Per Minute

    static dur qn, en, hn;

    fun void tempo(float beat) { // beat in BPM
        60.0 / beat => float spb; // seconds per beat
        spb::second => qn; // 1/4
        qn / 2.0 => en; // 1/8
        qn * 2.0 => hn; // 1/2
    }
}
```

```
// Play.ck
public class Play {

    fun void play(StkInstrument instrument, string notes[], dur durs[]) {
        for (0 => int i; i < notes.size(); i++)
            playnote(instrument, notes[i], durs[i]);
    }

    fun void playnote(StkInstrument instrument, string note, dur duration) {
        if (note != "--") {
            Std.mtof(s2n(note)) => instrument.freq;
            1 => instrument.noteOn;
        }
        duration => now;
        1 => instrument.noteOff;
    }

    fun int s2n(string name) {
        [21,23,12,14,16,17,19] @=> int notes[]; // A0,B0,C0,D0,E0,F0,G0
        name.charAt(0) - 65 => int base; // A=0,B=1,C=2,D=3,E=4,F=5,G=7
        notes[base] => int note;
        if (0 <= base && base <= 6) {
            if (name.charAt(1) == '#' || name.charAt(1) == 's') // sharp
                notes[base] + 1 => note;
            if (name.charAt(1) == 'b' || name.charAt(1) == 'f') // flat
                notes[base] - 1 => note;
        }
        else
            <<< "Illegal Note Name!" >>>;
        name.charAt(name.length()-1) - 48 => int oct; // 0, 1, 2, ..., 9
        if (0 <= oct && oct <= 9) {
            12 * oct +=> note;
            return note;
        }
        else
            <<< "Illegal Octave!" >>>;
    }

}
```

```
// melody.ck
BPM t;
t.qn => dur qn;
t.hn => dur hn;
t.en => dur en;
Play p;

[
"F4","G4","A4","F4",            "F4","G4","A4","F4",
"A4","Bb4","C5",                "A4","Bb4","C5",
"C5","D5","C5","Bb4","A4","F4", "C5","D5","C5","Bb4","A4","F4",
"F4","C4","F4","--",            "F4","C4","F4","--"
] @=> string melody[];

[
 qn,  qn,  qn,  qn,             qn,  qn,  qn,  qn,
 qn,  qn,  hn,                  qn,  qn,  hn,
 en,  en,  en,  en,   qn,  qn,  en,  en,  en,  en,   qn,  qn,
 qn,  qn,  qn,  qn,             qn,  qn,  qn,  qn
] @=> dur durs[];


Rhodey piano => dac;
p.play(piano, melody, durs);
```

```
// score.ck
BPM t;
t.tempo(150);
```

```
// starter.ck
Machine.add(me.dir() + "BPM.ck");
Machine.add(me.dir() + "Play.ck");
Machine.add(me.dir() + "score.ck");
```

9. 완성코드 1

```
Rhodey piano[4];
piano[0] => dac;
piano[1] => dac;
piano[2] => dac;
piano[3] => dac;

spork ~ p.play(piano[0], melody, durs);
t.qn * 8 => now;
spork ~ p.play(piano[1], melody, durs);
t.qn * 8 => now;
spork ~ p.play(piano[2], melody, durs);
t.qn * 8 => now;
spork ~ p.play(piano[3], melody, durs);
t.qn * 32 => now;
```

9. 완성코드 2

```
Rhodey piano0 => dac;
Rhodey piano1 => dac;
Rhodey piano2 => dac;
Rhodey piano3 => dac;

spork ~ p.play(piano0, melody, durs);
t.qn * 8 => now;
spork ~ p.play(piano1, melody, durs);
t.qn * 8 => now;
spork ~ p.play(piano2, melody, durs);
t.qn * 8 => now;
spork ~ p.play(piano3, melody, durs);
t.qn * 32 => now;
```

10. 완성코드

```
// score.ck
BPM t;
t.tempo(150);

Machine.add(me.dir() + "melody.ck");
t.qn * 8 => now;
Machine.add(me.dir() + "melody.ck");
t.qn * 8 => now;
Machine.add(me.dir() + "melody.ck");
t.qn * 8 => now;
Machine.add(me.dir() + "melody.ck");
t.qn * 32 => now;
```