Question 1  Consider the following Java program:

```java
class Zero {
    public int f (Zero zero) { return 0;}
}

class One extends Zero {
    public int m () { return 1;}
    public int f (One one) { return one.m();}
}

class Two extends One {
    public int m () { return 2;}
}

class Test {
    public static void main(String argv[]) {
        Zero zero = new Two();
        One one = new One();
        C;
    }
}
```

Say, for each case below, what happens if we replace C by the specified command (please, at least 1-2 lines for each case, explaining what happens at compile-time and, if compilation is successful, at run-time).

1. C = System.out.println(zero.m());
2. C = System.out.println(((One)zero).m());
3. C = System.out.println(one.f(zero));
4. C = System.out.println(one.f((One)zero));
5. C = System.out.println(((Two)one).m());

Question 2  Consider the following Java code.

```java
class A {
    int bin (A x) { return 1;}
}
class B extends A {
    int bin (A x) { return 3;}
    int bin (B x) { return 2;}
}

A aa=new A(); B bb=new B(); A ab=bb;
```

1. Which is the difference in the effect of the two method declarations in class B?
2. Show the value of the expression .bin(.) for each of the (nine) possible combinations of aa, bb and ab.
3. Change the definition of int bin(A) in class B in such a way that the value 2 is returned only when the two objects are instances of B, and 1 is returned otherwise.
Question 3  Give an example, analogous to that in course notes 5.3, which shows that allowing more generic types in fields of subclasses leads to an unsound type system. Use classes Rectangle and Cuboid as in course notes, and the following classes

```java
class Parent {
    Cuboid f;
}
class Heir extends Parent {
    Rectangle f;
}
```

and construct a class Break analogously to the example in course notes. Hint: the counterexample should work “the opposite way” w.r.t. that in course notes.

Question 4  Consider the following Java classes.

```java
class Parent {
    Parent m () { return this;}
}
class Heir extends Parent {
    int n () { return 2; }
}
```

and the expression `new Heir().m().n()`.

1. Is this expression well-typed? Motivate the answer, at least informally, or referring to the typing rules of MINIJAVA.

2. Would the evaluation of this expression get stuck? Motivate the answer, at least informally, or using the reduction rules of MINIJAVA.

3. Explain which is the problem with this example, and how the problem could be solved using a type system different from that of Java (and MINIJAVA).

Question 5  Consider the following Java code.

```java
class Parent{
}
class Heir extends Parent{
    int m () { return 1;}
}
class Client1 {
    int f1;
    int m (Heir x) { return x.m();}
    Parent n () { return new Parent();}
}
class Client2 {
    int f2;
    int m (Parent x) { return 2;}
    Heir n () { return new Heir();}
}
```

1. Consider an assignment `c1 = new Client2();` with `c1` of type `Client1`. Is the assignment well-typed in Java? Why?

2. Write the equivalent program in the language SOOC.

3. Is the (equivalent of the) previous assignment well-typed in SOOC? Why?