Sample Solution for Homework 11

Problem 1 Warm-Up on Objects (16 Points)

This is a warm-up exercise to get yourself familiar with objects (which you will implement in the second part of this homework).

For each of the following programs: (1) say whether the program can be safely evaluated or not, i.e., whether the evaluation will get stuck or produce a value. If it can be safely evaluated, provide the computed value; (2) determine whether the program is well-typed according to the typing rules given in figures 3 and 4. (Note that these rules do **not** yet allow for subtyping.) If the program is not well-typed, provide a brief explanation of the type error.

(i)

```
1 const x = {let f: 3};
2 const fun =
3   (y: {let f: Num, let g: Bool}) => (y.g = true, y);
4 fun(x).f
```

The program cannot be safely evaluated. The program would get stuck when `y.g = true` is executed because the object `x` passed to `fun` has no field `g`. The program is also not well-typed. The call to `fun` on line 4 passes an object of type `{let f: Num}`, which is not equal to `{let f: Num, const g: Bool}`, the expected type of `fun`'s parameter.

(ii)

```
1 const x = {let f: 3};
2 const fun =
3   (y: {let f: Num, const g: Bool}) => (y.f = 4, y);
4 fun(x).f
```

The program can be safely evaluated. The result value is `4`. However, it is not well-typed. The call to `fun` on line 4 still passes an object of type `{let f: Num}`, which is not equal to `{let f: Num, const g: Bool}`, the expected type of `fun`'s parameter. We need subtyping to make the program well-typed.

(iii)

```
1 const x = {let f: 3};
2 const fun = (y: {let f: Num}) => (y.f = 4, y);
3 fun(x).f
```

The program can be safely evaluated. The result value is `4`. The program is also well-typed.

(iv)

```
1  const x = {let f: 1, const g: true};
2  const y = {const f: false, let g: 2};
3  const z = true ? x : y;
4  z.f
```

The program can be safely evaluated. The result value is `1`. However, the program is not well-typed. The typing rule for conditional expressions demands that the type of `x` is equal to the type of `y`. However, they are not because they disagree on the type of the field `f`. We need subtyping to make this program well-typed.