IFC Challenges Solutions

Group 47 | Wenjun Tian | Yichen Li

Q1

```
l = h;
```

There is no restriction on the explicit flow from high variable to low variable. Or to say, there is even no typing system.

Q2

```
if (h) l = true; else l = false;
```

There is no restriction on the implicit flow from high variable to low variable. Everything is accepted except a direct assignment involving a high variable to a low one.

Q3

```
hatch = h;
l = declassify(hatch);
```

The value of the variable `hatch` can be extracted by wrapping it in `` `declassify() ``.

$$\frac{pc \sqsubseteq \Gamma(x)}{pc \vdash x = \texttt{declassify}(hatch);}$$

As we can see, the `disclassify()` function generally allows all flows from high/low to high/low.

Q4

```
let (x = h) in l = x;
```

$$\frac{pc \vdash c}{pc \vdash \texttt{let}\ x = e;\ \texttt{in}\ c}$$

According to the rule, we know that the expression `x = e` is not even checked by the typing system. Thus. we use `x = h` to create a explicit flow from high level variable `h` to low level variable `x`, and write the expression `c` as `l = x` to exploit the high level value of `h` from the low level variable `x`.

Q5

```
x = true;
l = true;
try {
    h_tt = h && l;
    h_ff = !(h || l);
    if (h_tt || h_ff) {
        skip;
    } else {
        throw;
    }
  } catch {
    while (x) {
        x = !x;
        l = !l;
    }
  }
```

Notice that by using `throw` in the `if-else` blocks, we can implicitly leak the the value of high level `if` statement to the low level `catch` block from one certain branch of `if-else`.

In our solution, the `if` statement checks if `h == l`. If true, then do nothing; else, jump to `catch` block , and invert the value of `l`.

The pseudo code is as follows:

```
if (h == l) skip; else l = !l;
```

Since there is no `==` operator, we construct the `h == l` statement as follows:

```
h_tt = h && l; //both true
h_ff = !(h || l); //both false
h_equals_l = h_tt || h_ff
```

If `h != l`, to invert the value of `l`, we construct this:

```
x = true; //low level variable
while (x) { // this block will be executed only once
        x = !x;
        l = !l;
    }
```

P.S.: the `l = true;` statement in the code is just an initialization of `l`, it can also be `l = false;`.