



```
toNat = \b -> if b 1 0
```

```
l1 = Nil
```

```
l2 :: List Bool
```

```
l2 = Nil
```

- $\text{exp}_1 = \lambda f.m f l1$
- $\text{exp}_2 = \lambda f.m f l2$
- $\text{exp}_3 = m \text{ isZero } l1$
- $\text{exp}_4 = m \text{ toNat } l2$
- $\text{exp}_5 = m \text{ double } l1$
- $\text{exp}_6 = m \text{ double } l2$

You should use the compilation technique that can deal with type declarations. So, you only have to determine the type of the functions in the program once (using  $\mathcal{W}$ , with consideration of the type declarations). Then you should build a new type assumption  $\mathcal{A}_1$  from the information you got by type-checking the program. Afterwards you only have to evaluate  $\mathcal{W}(\mathcal{A}_1, \text{exp}_i)$  for each of the six expressions  $\text{exp}_1, \dots, \text{exp}_6$ .

Here, the type assumption for `==` is `Int -> Int -> Bool`.