

Michael Aiello
0266204

CS 637 Project 2
(Unification Project)
Basic and “More Extensive Version”

Project Content

```
merge_bindings :: MyBindings -> MyBindings -> MyBindings
merge_bindings a [] = a
merge_bindings [] b = b
merge_bindings ((a,aelm):at) ((b,belm):bt)
  | genericLength (intersect a b) == 0 = ((a,aelm):at) ++ ((b,belm):bt)
  | genericLength (intersect a b) /= 0 && (aelm == belm || aelm == Nothing) = ((union a b),belm):bt)
  | genericLength (intersect a b) /= 0 && (belm == Nothing) = ((union a b),aelm):bt)
  | otherwise = error ("Can Not Unify " ++ (show a) ++ " to two different things!")
```

Project Content with explanation

```
merge_bindings :: MyBindings -> MyBindings -> MyBindings
-- function takes 2 lists of bindings, merges any overlaps it finds and makes sure the same thing isn't
-- bound to two different things

merge_bindings a [] = a
-- if we get a list and nothing, return the list

merge_bindings [] b = b
-- if we get nothing and a list, return the list
```

```

merge_bindings ((a,aelm):at) ((b,belm):bt)
-- split the input into
-- a, the list of strings unified to
-- aelm, the element that the list of string that the first binding binds to and
-- at, the rest of the bindings in the first list of bindings
-- b, the list of strings unified to
-- belm, the element that the list of string that the first binding binds to and
-- bt, the rest of the bindings in the second list of bindings

    | genericLength (intersect a b) == 0 = ((a,aelm):at) ++ ((b,belm):bt)
--find the length of the set of strings that exist in both a and b
--if there aren't any strings that exist in both
--then just return the two lists of bindings concatenated to each other, we didn't have to do any merging

    | genericLength (intersect a b) /= 0 && ( aelm == belm || aelm == Nothing) = (( (union a b),belm):bt)
--find the length of the set of strings that exist in both a and b
--if there are any strings that exists in both
-- and the elements we want to bind them to are the same
-- or a's element is nothing
--bind the union of the two sets to b's element and return it

    | genericLength (intersect a b) /= 0 && ( belm == Nothing ) = (( (union a b) ,aelm):bt)
--find the length of the set of strings that exist in both a and b
--if there are any strings that exists in both
--and b's element is nothing
--bind the union of the two sets to a's element and return it

    | otherwise = error ("Can Not Unify " ++ (show a) ++ " to two different things!")
--at this point we have realized that (aelm /= belm) and that is not allowed because
--you can not unify a string to two different elements
--so we print an error

```

This program considers general lists when checking the intersection, so there is no limitation on nesting as long as the bindings being checked are either lists or elements but not both. When I say “string” above referring to the list of items to bind to an element, it can be replaced with list to generalize.

Test Cases and output (these are the same test cases given in the PDF)

```
unify2lists [MyNum 1, MyNum 2, MyNum 3] [MyAtom "a", MyAtom "b", MyAtom "c"]  
[[["a"],Just (MyNum 1)],(["b"],Just (MyNum 2)),(["c"],Just (MyNum 3))]
```

```
unify2lists [MyNum 1, MyAtom "b", MyNum 3] [MyAtom "a", MyNum 2, MyAtom "c"]  
[[["a"],Just (MyNum 1)],(["b"],Just (MyNum 2)),(["c"],Just (MyNum 3))]
```

```
unify2lists [MyNum 1, MyNum 2, MyNum 3] [MyAtom "b", MyAtom "b", MyAtom "c"]  
Program error: Can Not Unify ["b"] to two different things!
```

```
unify2lists [MyNum 1, MyAtom "a", MyNum 3] [MyAtom "a", MyAtom "b", MyAtom "c"]  
[[["a","b"],Just (MyNum 1)],(["c"],Just (MyNum 3))]
```

```
unify2lists [MyNum 1, MyNum 2, MyNum 3] [MyNum 1, MyNum 4, MyNum 5]  
Program error: Unification: 3!=5
```

```
unify2lists [MyNum 1, MyNum 2, MyNum 3] [MyNum 1, MyAtom "a", MyNum 5]  
Program error: Unification: 3!=5
```

```
Main> unify2lists [MyAtom "a", MyAtom "b", MyAtom "c"] [MyAtom "b", MyAtom "c", MyAtom "a"]  
[[["a","b","c"],Nothing]]
```

```
unify2lists [MyNum 1, MyNum 4, MyNum 3] [MyNum 1, MyNested[MyNum 4], MyNum 3]  
Program error: Cannot unify number 4 with list [MyNum 4]
```

```
unify2lists [MyNum 1, MyNum 2, MyNested[MyNum 3]] [MyNum 1, MyAtom "a", MyNested[MyNum 3, MyNum 1]]  
Program error: Unequal length lists [MyNum 3] and [MyNum 3,MyNum 1]
```

```
unify2lists [MyNum 1, MyNested[MyNum 2], MyNum 3] [MyAtom "a", MyNested[MyAtom "b"], MyAtom "c"]
```

```
[(["a"], Just (MyNum 1)), (["b"], Just (MyNum 2)), (["c"], Just (MyNum 3))]
```

```
unify2lists [MyNum 1, MyNested[MyAtom "b"], MyNested[MyNested[MyNum 3]]] [MyAtom "a", MyNested[MyNum 2],  
MyNested[MyNested[MyAtom "c"]]]
```

```
[(["a"], Just (MyNum 1)), (["b"], Just (MyNum 2)), (["c"], Just (MyNum 3))]
```

Note on the ugliness of the test cases: I couldn't get the nice_unify2 function to work properly, even on the simplest of cases with the original merge_bindings a b = a ++ b definition.