```julia
function find_pairs(numbers, target_sum)
    pairs = []
    seen = Set()
    for num in numbers
        complement = target_sum - num
        if complement in seen
            push!(pairs, (complement, num))
        end
        push!(seen, num)
    end
    return pairs
end

# Example usage
numbers = [1, 3, 8, 12, 7, 11, 9, 4, 2, 10, 5]
target_sum = 12
println(find_pairs(numbers, target_sum))
```

```
Any[(1, 11), (3, 9), (8, 4), (2, 10), (7, 5)]
julia>
```

```haskell
import Data.List (nub)

findPairs :: [Int] -> Int -> [(Int, Int)]
findPairs lst sum = nub [(x, y) | x <- lst, y <- lst, x < y, x + y == sum]

-- Example usage
main :: IO ()
main = do
  let numbers = [1, 3, 8, 12, 7, 11, 9, 4, 2, 10, 5]
  let targetSum = 12
  print $ findPairs numbers targetSum
  print $ findPairs numbers targetSum + 1
```

Interpret Leav 24m on 19:37:35, 11/21

Interpret 11s on 20:02:01, 11/21

```
GHCi, version 9.0.2: https://www.haskell.org/ghc/  :? for help
Loaded GHCi configuration from /home/runner/CalculatingLovelyProgrammingmacro/.ghci
[1 of 1] Compiling Main             ( Main.hs, interpreted )
Ok, one module loaded.
> [(1,11),(3,9),(4,8),(2,10),(5,7)]
>
```

```julia
function find_all_pairs(numbers)
    results = Dict()
    for i in 1:length(numbers)
        for j in (i+1):length(numbers)
            sum = numbers[i] + numbers[j]
            if !haskey(results, sum)
                results[sum] = []
            end
            push!(results[sum], (numbers[i], numbers[j]))
        end
    end
    return results
end

# Example usage
numbers = [1, 3, 8, 12, 7, 11, 9, 4, 2, 10, 5]
println(find_all_pairs(numbers))
```

Interpret 16s on 20:00:27, 11/21

```
Dict{Any, Any}(5 => Any[(1, 4), (3, 2)], 16 => Any[(12, 4), (7,
 9), (11, 5)], 20 => Any[(8, 12), (11, 9)], 12 => Any[(1, 11),
(3, 9), (8, 4), (7, 5), (2, 10)], 8 => Any[(1, 7), (3, 5)], 17
=> Any[(8, 9), (12, 5), (7, 10)], 23 => Any[(12, 11)], 19 => An
y[(8, 11), (12, 7), (9, 10)], 22 => Any[(12, 10)], 6 => Any[(1,
 5), (4, 2)], 11 => Any[(1, 10), (3, 8), (7, 4), (9, 2)], 9 =>
Any[(1, 8), (7, 2), (4, 5)], 14 => Any[(3, 11), (12, 2), (9, 5)
, (4, 10)], 3 => Any[(1, 2)], 7 => Any[(3, 4), (2, 5)], 4 => An
y[(1, 3)], 13 => Any[(1, 12), (3, 10), (8, 5), (11, 2), (9, 4)]
, 15 => Any[(3, 12), (8, 7), (11, 4), (10, 5)], 21 => Any[(12,
9), (11, 10)], 10 => Any[(1, 9), (3, 7), (8, 2)], 18 => Any[(8,
 10), (7, 11)])
julia>
```

Main.hs > main

```haskell
import qualified Data.Map as Map

findAllPairs :: [Int] -> Map.Map Int [(Int, Int)]
findAllPairs lst = foldl insertPair Map.empty [(x, y) | x <- lst, y <- lst, x < y]
  where
    insertPair m (x, y) = Map.insertWith (++) (x + y) [(x, y)] m

-- Example usage
main :: IO ()
main = do
  let numbers = [1, 3, 8, 12, 7, 11, 9, 4, 2, 10, 5]
  print $ findAllPairs numbers
  print $ findAllPairs (reverse numbers)
```

Interpret □□□ ghc: signal: 15 — 37s on 19:36:56, 11/21

Interpret Leav — 24m on 19:37:35, 11/21

Interpret □□□ — 24s on 20:02:01, 11/21

Interpret — 15s on 20:02:26, 11/21

```
GHCi, version 9.0.2: https://www.haskell.org/ghc/  :? for help
Loaded GHCi configuration from /home/runner/CalculatingLovelyProgrammingmacro/.ghci
[1 of 1] Compiling Main             ( Main.hs, interpreted )
Ok, one module loaded.
ghci> fromList [(3,[(1,2)]),(4,[(1,3)]),(5,[(2,3),(1,4)]),(6,[(2,4),(1,5)]),(7,[(2,5),(3,4)]),(8,[(3,5),(1,7)]),(9,[(2,7),(4,5),(1,8)]),(10,[(2,8),(3,7),(1,9)]),(11,[(2,9),(4,7),(3,8),(1,10)]),(12,[(5,7),(2,10),(4,8),(3,9),(1,11)]),(13,[(5,8),(2,11),(4,9),(3,10),(1,12)]),(14,[(5,9),(2,12),(4,10),(3,11)]),(15,[(5,10),(4,11),(7,8),(3,12)]),(16,[(5,11),(4,12),(7,9)]),(17,[(5,12),(7,10),(8,9)]),(18,[(7,11),(8,10)]),(19,[(9,10),(7,12),(8,11)]),(20,[(9,11),(8,12)]),(21,[(10,11),(9,12)]),(22,[(10,12)]),(23,[(11,12)])]
ghci>
```