```go
package main

import (
  "fmt"
  "strings"
)

var ()

func LUDecomposition(matrix1 [][]float64) (UMatrix, LMatrix [][]float64) {
  fmt.Println(strings.Repeat("#", 15))
  fmt.Println("Starting LU decomposition.")

  numberOfMisToCreate := len(matrix1) - 1 //Number of Mi matrices we will need
to find.
  matrixSize := len(matrix1)              //Size of the Mi matrix

  //Initializa LMatrix
  LMatrix = InitializeMatrixWithZeros(matrixSize, matrixSize)

  //We wont initialize UMatrix since it will be stored in matrix1.

  //Populate UMatrix and LMatrix values
  for i := 0; i < numberOfMisToCreate; i++ {
    MMatrix := createPivotMatrixM(matrix1, i)
    matrix1, _ = MultiplyMatrices(MMatrix, matrix1)
    if i == 0 {
      LMatrix = generateLiMatrixFromUiMatrix(MMatrix)
    } else {
      MInverse := generateLiMatrixFromUiMatrix(MMatrix)
      LMatrix, _ = MultiplyMatrices(LMatrix, MInverse)
    }
    fmt.Printf("\n")
  }

  fmt.Println("Finished LU decomposition")
  //We return matrix1 in place of UMatrix since matrix1 actually stores
```

```
    UMatrix. Also, we keep UMatrix on the signature to keep the function easy to
    understand and use.
37    return matrix1, LMatrix
38  }
39
40  func solutionViaLUDecomposition(c configuration) (res [][]float64) {
41    U, L := LUDecomposition(c.matrixA)
42    Lstring := CreateMatrixString(L)
43    Ustring := CreateMatrixString(U)
44
45    //Escrevendo em arquivo
46    Pw(OUTPUT_FILE_PATH, "Matriz L encontrada\n")
47    Pw(OUTPUT_FILE_PATH, Lstring)
48    Pw(OUTPUT_FILE_PATH, "Matriz U encontrada\n")
49    Pw(OUTPUT_FILE_PATH, Ustring)
50
51    res1 := forwardSubstitution(L, c.vectorB)
52    res2 := backwardsSubstitution(U, res1)
53    res2String := CreateMatrixString(res2)
54    //Escrevendo resultado final
55    Pw(OUTPUT_FILE_PATH, fmt.Sprintf("Resultado final:\n%s\n", res2String))
56    return res2
57  }
58
```