# Solutions to Exercise Sheet 1

## Exercise 1 - Metrics

### 1.1 Lines of Code Metrics

i $LOC_{tot} = 74$
$LOC_{ne} = 74 - 10 = 64$
$LOC_{pars} = 64 - 15 = 49$

ii Example Haskell-Code for contrasting the given MyQuickSort.java:

```haskell
module MyQuickSort where

-- This Code is self-Documenting
quicksort :: Ord a => [a] -> [a]
quicksort []     = []
quicksort (x:xs) = smaller ++ [x] ++ bigger
  where
  smaller = quicksort [y | y <- xs, y <= x]
  bigger  = quicksort [y | y <- xs, y >  x]
```

$LOC_{parsH} = 9 - 2 = 7$
So there is $LOC_{pars}$ with 49 as well as $LOC_{parsH}$ with 7 (Order of magnititude: $n^2$ *vs* $n!$). These are obviously two entirely different Programs, yet they are semantically equivalent in that they offer an interface to a function capable of sorting a List of items with a Quicksort-Algorithm.

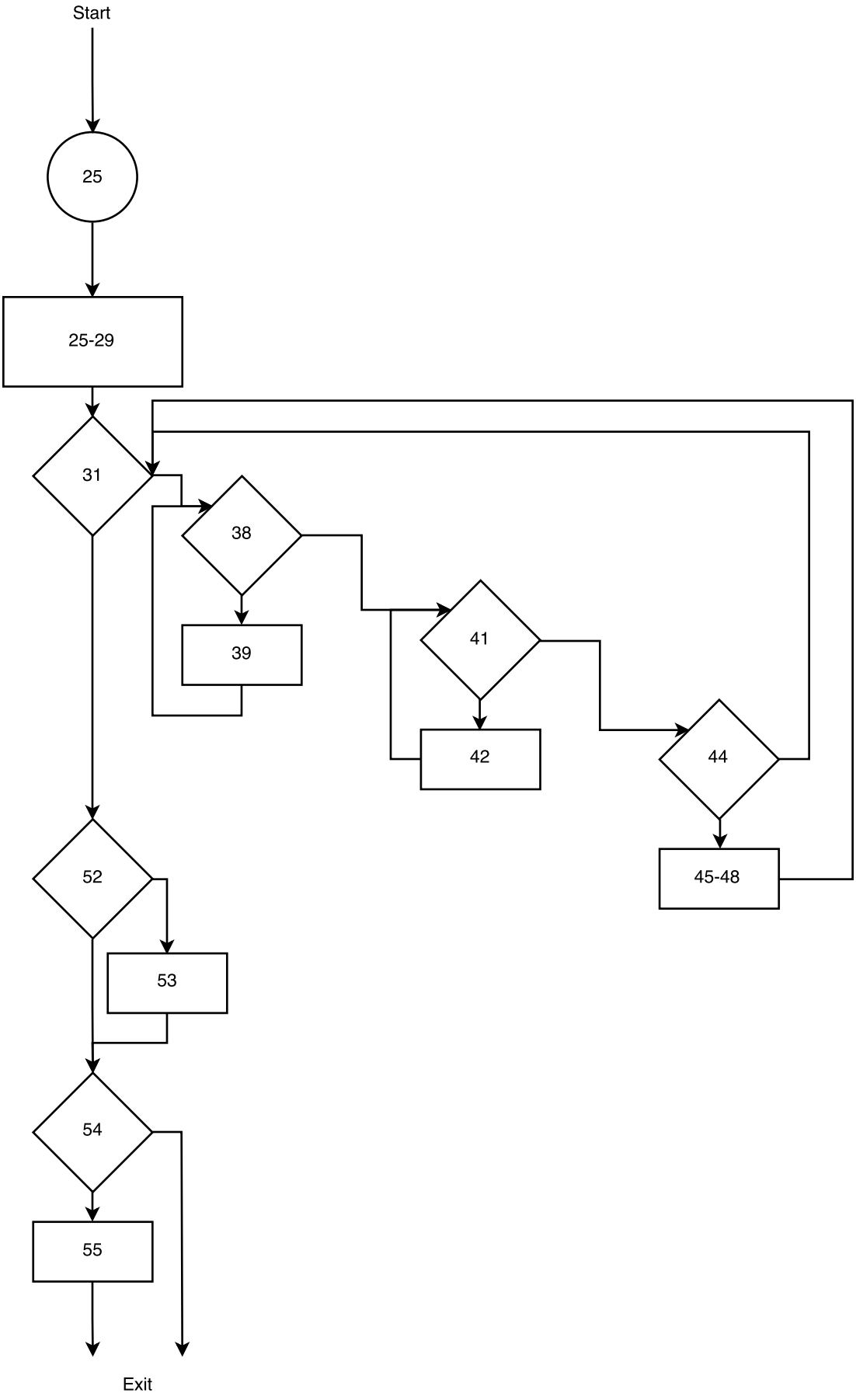
In this case the recognized Pattern is to use a library or tool requiring a lot of (hardcoded) configuration, where a simpler one would clearly suffice. That way you would have a lot of managing / organizing / configurating overhead, which can be overblown to the fullest if wanted (resulting in an LOC of literally any number you wish).

iii Metrics:
$LOCtot = 178$
$LOCne = 136$
$LOCpars = 120$
It basically configures all the GUI-Elements (like buttons etc.), their Positions, Sizes, how they should behave when resizing the window or when being clicked, and more. The Program itself is a simulator for AI-Ants to find paths in a generated Maze, the whole Project written in C++.

## 1.2 Cyclomatic Complexity

1. $p = 3$
   $n = 13$
   $e = 17$
   $v(G) = 17 - 13 + 3 = 7$
   The CFG will be on the last page.

2. Junction points in a CFG do not alter the cyclomatic complexity as it adds an edge for each node.


Start
25
25-29
31
38
39
41
42
44
45-48
52
53
54
55
Exit