Part 1.1

The assembly program implements the iterative method to find the square root of a give number. It uses basic subroutines to fulfill such task. Using subroutines like LOOP and ENDLOOP, the overall procedure is easy to follow as it is designed with a logic similar to the original code provided in the lab questions.

Part 1.2

The assembly program implements the recursive method to find the square root of a given number. It uses subroutines, conditional branching, and stack to fulfill the task. It follows the general procedure given in the lab question and uses recursive branching to complete the calculation. Before branching to SqrtRecur, it pushes values on registers 0 – 3 and the value of lr into the stack to save the values of the variables so that they won't be lost at the next call to SqrtRecur.

Part 2

The assembly program uses the procedure given in the lab code to complete the calculation of the norm to a given vector. To fulfill the task, it uses multiple subroutines and branching. Names for loops such as WHILELOOP and FORLOOP and their contents are designed to mimic the behavior in the lab code, so that the assembly code will be easy to follow and similar to the lab code in the aspect of overall flow and logic.

Part 3

The assembly program centers an array so that its average is 0 by using multiple subroutines and branching. Similar to Part2, it also uses loop names that are related to code given in the lab question. Conditional branching is used to implement the check of each loop.

Part 4

The assembly program implements the selection sort algorithm using multiple subroutines, branching and nested loops. Names for the loops such as OUTERLOOP and INNERLOOP are related to the code given in lab question. The pointer variable in the original code is implemented in assembly using pseudo command (such as LDR R1, =ARRAY).

The program uses too much registers for variables, and can thus be optimized – using less variables by finding a way to use one register as multiple variables.