Exercise 1:

(iii) From the pseudo-code provided in the class, we can get that:

1. For $selection\_sort$, if the length of the list is n, then we need to go through the loop for n times. Thus, we need:
    a. n steps for checking if the list is empty
    b. On average, searching through the list to find the smallest value needs $\frac{1}{2}\left(n+\frac{n(n+1)}{2}\right)=\frac{(n+3)n}{4}$ steps
    c. n steps to add the smallest value into the new list
    d. n steps to delete the smallest value in the old list

    Also, we need 1 step to go out the loop. So in total, we need $\frac{(n+3)n}{4}+1+3n$ steps.

2. For $Mergesort$, if the length of list is n, then:
    a. For the division part, we need $n-1$ steps to separate the list into single element
    b. For merge part, since the depth of the division in the first part is $logn$, we need to deal with the whole list for $logn$ times. In each time, we need to compare between different sublists for n times and add the smaller value into a new list for n times. So in total, we need $2n * logn$ in this part.

    So for $Mergesort$, the total number of steps is $2n * logn + n - 1$.

To find the proper t, we let the steps of two sorting algorithms be equal.

$$\text{i.e. } 2n * logn + n - 1 = \frac{(n+3)n}{4} + 1 + 3n$$

Finally, we get that $n = 26.54$, so $t = 26$.

For the experimental value, see the output of the program.