1. In the list below, $i$ passes of the selection sort algorithm have been completed, and the double bar separates
the sorted part of the list from the unsorted part.

<table>
<thead>
<tr>
<th>L</th>
<th>sorted</th>
<th>unsorted</th>
</tr>
</thead>
</table>

(a) `get_index_of_smallest(L, i)` works by comparing pairs of items from the unsorted section. If
there are $n$ items in $L$, when `get_index_of_smallest(L, i)` is executed, how many pairs of items
are compared? (Your answer should be a formula involving $n$ and $i$.)

$$n - i - 1$$

(b) For function `get_index_of_smallest(L, i)`, is there a worst case and a best case?

No

(c) In terms of the number of items in the unsorted section, does `get_index_of_smallest` have constant
running time, linear running time, quadratic running time, or some other running time?

(a) constant  (b) linear  (c) quadratic  (d) something else

(d) In function `selection_sort`, the first time that function `get_index_of_smallest` is called, $i$ is
0; the second time, $i$ is 1; and so on. What value does $i$ have the last time that function
`get_index_of_smallest` is called?

$$n - 1$$

(e) For the call `selection_sort(L)`, write a formula expressing how many comparisons are made
during all the calls to `get_index_of_smallest`.

$$\sum_{i=0}^{n-1} (n-i) = \frac{n(n-1)}{2}$$

(f) In terms of the length of the list, does `selection_sort` have constant running time, linear running
time, quadratic running time, or some other running time?

(a) constant  (b) linear  (c) quadratic  (d) something else