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.

$$i$$

<table>
<thead>
<tr>
<th>L</th>
<th>sorted</th>
<th>unsorted</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = \text{len}(L)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</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 function involving $n$ and $i$.)

Same as: How many times does the loop execute?

$j \rightarrow 0$ to $n-1$

$L \rightarrow (n-1) - (i+1) + 1$

$L \rightarrow 1 - i - 1$

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

No, same. What about # of assignments?

Best: $0$ in loop + $1$ outside

Worst: $n-2-1 \cdot n$ loop + $1$

(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`.

\[
\text{sum of } 1 + 2 + \ldots + n-1 \approx \frac{n(n-1)}{2}
\]

(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