Prep 11 quiz: Recursive Sorting Algorithms

Due Nov 27 at 10am **Points** 5 **Questions** 5
Available Nov 22 at 9am - Nov 27 at 10am 5 days **Time Limit** None
Allowed Attempts Unlimited

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	10 minutes	5 out of 5

ⓘ Correct answers are hidden.

Score for this attempt: **5** out of 5
Submitted Nov 26 at 3:07pm
This attempt took 10 minutes.

Question 1 — 1 / 1 pts

The two main iterative sorting algorithms we study in CSC108 are **selection sort** and **insertion sort**. Select the correct worst-case running time of these algorithms from the options below (*n* represents the length of the input list to be sorted).

- O(n)
- O(n log n)
- ◉ O(n^2)
- O(n^3)
- O(log n)

Question 2

1 / 1 pts

The two main recursive sorting algorithms we'll study in this course are **mergesort** and **quicksort**, which are both *divide-and-conquer* algorithms. Select the phases of the general divide-and-conquer sorting approach in the correct order below.

Step 1	Split the input into two or mc
Step 2	Recurse on each part separat
Step 3	Combine the results of the pı

Question 3

1 / 1 pts

Select all of true statements about the interface of `_merge` (the main helper of the mergesort algorithm).

- ☐ _merge will work as long as at least one of its input lists to be sorted.
- ☑ _merge requires all of its input lists to be sorted.
- ☐ _merge requires that all of its input lists have the same length.
- ☐ _merge may mutate its inputs.
- ☑ _merge returns a new list that is sorted.
- ☐ _merge takes any number of lists as inputs.
- ☐

_merge requires that all of the elements of the first input list are <= all of the elements in the second input list.

Question 4

1 / 1 pts

Select all of true statements about the interface of `_partition` (the main helper of the quicksort algorithm).

- ☐ _partition returns two lists of equal length.
- ☑ _partition returns two lists whose elements are from its input list.
- ☐ _partition takes two lists as its arguments.
- ☑ _partition returns two lists, where every element in the first list is < every element in the second list.
- ☑ _partition takes a list and object as its arguments.
- ☐ _partition mutates its input list.

Question 5

1 / 1 pts

The versions of mergesort and quicksort that we've provided in the readings are non-mutating, and return a new list that's the sorted version of the original.

Below is a small modification of our mergesort algorithm to make it mutate its input list. Does this implementation succeed, and why or why not?

```
def mergesort(lst: list) -> None:
    """MUTATE <lst> so that it is sorted.
    """
    if len(lst) < 2:
        # Do nothing
```

```
        pass
    else:
        # Divide the list into two parts, and sort them recursively.
        mid = len(lst) // 2
        left = lst[:mid]
        right = lst[mid:]
        mergesort(left)
        mergesort(right)

        # Merge the two sorted halves.
        # Assume _merge has the exact same behaviour as before.
        lst = _merge(left, right)
```

This version **DOES** satisfy its docstring, because `left` and `right` are correctly sorted, and the last line merges them together and stores the result in the input list.

This version **DOES NOT** satisfy its docstring, because the two recursive calls `mergesort(left)` and `mergesort(right)` need to return something in order to have an effect.

This version **DOES** satisfy its docstring, but the last line doesn't need to assign to `lst`. Since `left` and `right` are aliases of the left and right halves of the input list, calling `_merge(left, right)` already sorts `lst`, and so the assignment is unnecessary.

This version **DOES NOT** satisfy its docstring, because assigning to `lst` in the last line only changes what a local variable refers to, and not the contents of input list itself.

Quiz Score: **5** out of 5