CS 101 Fall 2022 - Quiz 3-B

10/11/2022 - 20 Minutes Name: ID number:

---

1. (6 points) Multiple Choices

Each of the following questions has **one or more** correct answer(s). Please choose all the correct answers. If your answer is a non-empty strict subset of the correct answers, you will receive 1pt. Write your answers in the table below.

(a)	(b)	(c)

(a) (2') Which of the following sorting algorithms run in $O(n^2)$ time?

A. Insertion-sort B. Merge-sort C. Bubble-sort D. Quick-sort

(b) (2') As for time complexity, which of the following statements are true?

A. Insertion-sort has the best time complexity on a sorted array among all sorting methods.

B. Quick-sort runs in $O(n \log n)$ time in worst case if the pivots are chosen via the 'median-of-three' method (that is, to choose the median of $\{a_l, a_m, a_r\}$ as the pivot when partitioning the subarray $\langle a_l, \cdots, a_r \rangle$, where $m = \lfloor (l + r)/2 \rfloor$).

C. Bubble-sort, if modified with certain tricks, could run in $\Theta(n)$ time if there are $O(n)$ inversions.

D. Merge-sort has a worst-case runtime that is asymptotically better than the worst-case runtime of quick-sort.

(c) (2') Which of the following statements are true?

A. A sorting algorithm is *stable* if its worst-case time complexity is the same as its best-case time complexity.

B. Insertion-sort is stable.

C. Merge-sort requires $\Theta(\log n)$ extra space when sorting an array of n elements.

D. Quick-sort only uses $O(1)$ extra space.

2. (2 points) Inversions

Suppose we are performing merge-sort on an array. At a certain step, we need to merge two sorted subarrays $\langle a_1, a_2, a_3, a_4, a_5 \rangle$ and $\langle b_1, b_2, b_3, b_4, b_5, b_6 \rangle$ into one. Assume that these elements are distinct. Suppose the result is

$$\langle b_1, b_2, a_1, a_2, a_3, b_3, a_4, b_4, a_5, b_5, b_6 \rangle.$$

From this you can infer that the number of inversions in the original array is at least ______________.

3. (6 points) Merging Linked-lists

Liu Big God has found an interestingly designed linked-list library in his grandfather's computer. The library was developed over 30 years ago, and provides interfaces that are quite different than what we see in lectures. It mainly contains a `List` class, which represents a singly-linked list (assuming the data it stores are `int`s), with the following operations supported (suppose `l` is a `List` and `x` is an `int`).

- `cons(x, l)` returns a `List` obtained from `l` by inserting `x` to the beginning of it.
- `l.car()` returns the first element of `l`. Runtime-error if `l` contains no elements.
- `l.cdr()` returns a `List` consisting of all the elements of `l` except the first. Runtime-error if `l` contains no elements.
- `l.null()` returns `true` if `l` contains no elements, `false` otherwise.
- `List::nil` is a `List` with no elements.

Curious about how this `List` works, Liu Big God is trying to perform merge-sort on it (in ascending order). Please help him with the `merge` procedure, which merges two sorted `List`s into one.

```
List merge(const List &x, const List &y) {
  if (x.null())
    ______________;
  if (y.null())
    ______________;
  int xh = x.car(), yh = y.car();
  if (xh < yh)
    return _______________________________;
  else
    return _______________________________;
}
```

(a) (2') Fill in the first two blanks, which handle the cases where one of the given `List`s is empty.

(b) (4') Fill in the rest two blanks, which finishes the work in a **recursive** way.

Please note that:

- One statement for each blank.
- Only the five operations listed above are available. It is not allowed to use other operations like `push_front`, `pop_front` or `insert_after`.
- All the implementation details of `List` are `private`. Direct access to nodes, data or pointers will lead to compile error.
- You don't need to worry about the time complexity of `cons` and `cdr`.

4. (2 points) Guess the average score ($\in [0, 16]$) of this quiz.

4. ______________