COS 226 Algorithms and Data Structures Spring 2016

Midterm Exam

This midterm has 7 questions for a total of 66 points.

You have 80 minutes. The exam is closed book, and no calculators or other electronic devices are allowed. You may use one 8.5×11 page (one side) of notes in your own handwriting.

Name:

NetID:

Precept:

Room in which you’re taking the exam:

P01	Th 9–9:50am	Maia Ginsburg
P02	Th 10–10:50am	Shivam Agarwal
P02A	Th 10–10:50am	Marc Leef
P03	Th 11–11:50am	Maia Ginsburg
P03A	Th 11–11:50am	Ming-Yee Tsang
P04	Th 12:30pm–1:20pm	Miles Carlsten
P05	Th 1:30pm–2:20pm	Sergiy Popovych
P06	F 10–10:50am	Andy Guna
P07	F 11–11:50am	Andy Guna
P07A	F 11–11:50am	Harry Kalodner
P99	M 7:30–8:20pm	Andy Guna

Write and sign:

“I pledge my honor that I have not violated the Honor Code during this examination.”

Optional: mark how you feel at the beginning of the test... ...and at the end

1. Heaps (6 points).

Here's an array representation of a min-heap.

```
   i    0  1  2  3  4  5  6  7  8  9 10 11
a[i]    _  C  G  H  K  M  L  U  V  R  Q  P
```

Show the resulting array after performing each of the following operations.

(a) Insert **E**.

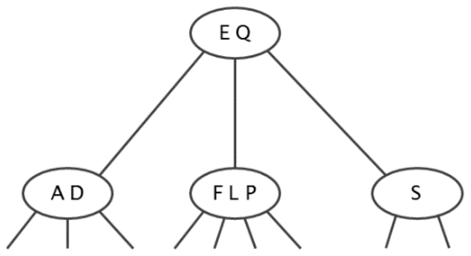
i	0	1	2	3	4	5	6	7	8	9	10	11	12
a[i]	__												

(b) Delete the minimum, starting from the original heap (i.e, <u>without</u> performing (a) first).

i	0	1	2	3	4	5	6	7	8	9	10
a[i]	__										

2. Balanced search trees (6 points).

Recall that inserting into a 2-3 tree temporarily creates 2-3-4 trees. Here is such a 2-3-4 tree.



(a) Draw the corresponding left-leaning red-black tree. Mark red links with 'r'. (Note that it may violate some of the conditions for a valid LLRB.) If you end up drawing several trees, make sure you circle your final answer.

(b) Draw the left-leaning red-black tree that results after the insert has been completed.

(c) [Extra credit, 1 point] Show one possible order in which these 8 keys could have been inserted, starting from an empty tree, so that the result would be the 2–3–4 tree shown on the previous page.

3. Union-find (10 points).

(a) In each case, write *yes* in the right column if the array **`a[i]`** could represent the state of quick find after some sequence of union operations, and *no* otherwise.

Quick find

Possible?

i	0	1	2	3	4	5	6	7	8
a[i]	3	1	4	4	4	4	3	4	4

i	0	1	2	3	4	5	6	7	8
a[i]	1	1	1	1	4	4	4	4	4

(b) In each case, write *yes* in the right column if the array **`a[i]`** could represent the state of weighted quick union after some sequence of union operations, and *no* otherwise.

Weighted quick union

Possible?

i	0	1	2	3	4	5	6	7	8
a[i]	1	8	8	5	8	5	7	7	7

i	0	1	2	3	4	5	6	7	8
a[i]	2	1	1	1	4	4	3	5	4

i	0	1	2	3	4	5	6	7	8
a[i]	0	4	3	3	5	1	3	3	2

4. Collections (8 points).

The class **`Mystery`** implements a collection that we've studied in class:

```
public class Mystery<Item> {
      private Stack<Item> s;

      public Mystery() {
            s = new Stack<Item>();
      }

      public boolean isEmpty() {
            return s.isEmpty();
      }

      public void add(Item item) {
            s.push(item);
      }

      public Item remove() {
            assert !s.isEmpty();
            Item x = s.pop();
            if (s.isEmpty())
                  return x;
            Item y = remove();
            s.push(x);
            return y;
      }
}
```

(a) Suppose we make 3 **`add()`** calls to a newly constructed **`Mystery`** object, followed by a **`remove()`**. Write the sequence of **`s.push()`** and **`s.pop()`** operations invoked in executing **`remove()`**. [For example, you might write something like **`push, pop, push`**.]

(b) Which data type does **`Mystery`** implement?

(c) What's the order of growth of the worst-case running time of **`add()`** on a **`Mystery`** object with N items? What about **`remove`**? Assume a linked-list implementation of **`Stack`**.

add: O($\boxed{}$)   remove: O($\boxed{}$)

5. Sorting (12 points).

For each of these statements about sorting algorithms, state whether it is true or false.

(a) Insertion sort has optimal best-case running time. ________

(b) In any given array, bottom-up mergesort will compare the same pairs of keys as top-down mergesort; it just does the comparisons in a different order. ________

(c) Selection sort compares each pair of keys at most once. ________

(d) Mergesort with cutoff to insertion sort for subarrays of size ≤ 10 items has $O(n \log n)$ worst-case running time (where n is the size of the array). ________

(e) Mergesort with cutoff to insertion sort for subarrays of size $\leq \log n$ items has $O(n \log n)$ worst-case running time (where n is the size of the array). ________

(f) For arrays where every key has at least one duplicate, 3-way quicksort runs in linear time. ________

(g) One reason heapsort can be slower than quicksort in practice is that it makes poor use of cache. ________

(h) We can sort an array by date by first sorting it by day, then by month, then by year, but only if we use a stable sorting algorithm. ________

6. Memory (12 points).

(a) Consider the following separate-chaining hash table implementation. Only the variables are shown, methods are not.

```
public class SeparateChainingHashST<Key, Value> {

      private static class Node<Key, Value>  {
            private Key key;
            private Value val;
            private Node next;
            ...
      }

      private int M;                 // length of st array
      private Node[] st;             // array of chains
      ...
}
```

Using the 64-bit memory cost model from lecture and the textbook, estimate the number of bytes of memory consumed by the hash table as a function of M (which is the length of the array **st**), and the number of keys N. Express your answer in tilde notation. Ignore the memory consumed by the keys and values themselves.

Hint: since you're allowed to use both M and N in your answer, you don't need to worry about expressing M in terms of N (i.e., no need to take into account the average chain length).

~ $\boxed{}$ bytes

(b) Now consider the following linear-probing hash table implementation. As before, only the variables are shown.

```
public class LinearProbingHashST<Key, Value> {
      private int L;                // length of keys and vals arrays
      private Key[] keys;           // array of keys
      private Value[] vals;         // array of corresponding values
      ...
}
```

Estimate the number of bytes of memory consumed by this implementation as a function of the array length L and the number of keys N. Express your answer in tilde notation. Ignore the memory consumed by the keys and values themselves.

~ $\boxed{}$ bytes

(c) After a certain number of keys N have been inserted into each hash table implementation, it is observed that they have both consumed the same amount of memory. It is also observed that the average chain length in the separate chaining implementation is 4.0. Based on this information, calculate the fraction of the linear-probing hash table that's filled up.

(Write a number between 0 and 1, either as a decimal or as a fraction.)

$\boxed{}$

7. Design (12 points).

Call two strings equivalent if one is an anagram of the other, that is, one string can be turned into the other by rearranging the characters. An equivalence class is a set in which any pair is equivalent. For example, in the array `["aaa", "aab", "aba"],` there are two equivalence classes since the latter two strings are equivalent but neither is equivalent to the first string.

Given an array of strings, design an algorithm to find the number of equivalence classes among them. Write your answer in words, rather than code or pseudocode. But be precise about your data structures and the steps of your algorithm.

Analyze the order-of-growth running time of your algorithm. For full credit, it should run in time $O(NM(\log N + \log M))$ in the worst case, where N is the length of the array and M is the maximum length of the strings.

Use this space for scratch. And don’t forget to write your info on the front page!

Problem	Score
1	
2	
3	
4	
5	
6	
7	
Total	