111-1 PDSA Midterm

Oct. 24, 2022

學號：________________

姓名：________________

1. a. (5%) Please define what a **stable sort** is.
   b. (5%) Explain how **insertion sort** achieves the stable property.
   c. (5%) Explain how **mergesort** achieves the stable property.

2. (15%) (a) Please describe the procedures of the 'Knuth Shuffle' algorithm that produces a uniformly random permutation of the input array. (b) Prove that the 'Knuth Shuffle' algorithm is unbiased, so that every permutation is equally likely.

3. (10%) Which of the following id[] array(s) could be the result of running the weighted quick union algorithm on a set of 10 items? Check all that apply.

```
(a) 8 0 2 3 0 8 5 6 3 5
(b) 9 5 4 6 4 4 4 4 4 5
(c) 4 4 6 4 5 2 4 4 4 6
(d) 0 2 2 3 4 5 0 7 9 9
(e) 6 9 9 6 0 2 9 0 9 9
```

   Recall: our weighted quick union algorithm uses union by size (number of nodes), not union by height.

4. (15%) Please implement the three functions ('isEmpty', 'enqueue' and 'removeFirst') of CircularQueue.java with the following API:

```
public class CircularQueue <Item> implements Iterable {
   private Node last;
   private class Node{
      Item item;
      Node next;
   }
   ...
   public boolean isEmpty() { // Is the circular queue empty?
   // Provide your code here
   }
   public void enqueue(Item item) { // Add item to the end of the list
   // Provide your code here
   }
   public Item removeFirst() { // Remove item from the beginning of the
list
   // Provide your code here
   }
   ...
}
```

In a circular queue, no links are **null** and the value of last.next is first whenever the list is not empty. Keep only one Node instance variable (last).

5. (15%) Write a recursive function that converts **infix expressions** to **postfix**.

```
public String infix2postfix (String e) {
    private String s;
    // Provide your code here
    return s;
}
```

Example of input: (1 + ((2 + 3) * (4 * 5)))

Example of output: 1 2 3 + 4 5 * * +

Note: All the numbers, operators, and brackets are separated by 'single space', and each bracket contains one operator and two operands only.

6. (10%) Given an array of points, **which of the following approaches** would be *least useful* for removing duplicate points? (Assume the point data type has the following three orders: (1) A natural order that compares by x-coordinate and breaks ties by y-coordinate; (2) One comparator that compares by x-coordinate; and (3) Another comparator that compares by y- coordinate.)

a. quicksort by natural order;

b. quicksort by x-coordinate, mergesort by y-coordinate;

c. mergesort by x-coordinate, quicksort by y-coordinate;

d. mergesort by x-coordinate, mergesort by y-coordinate.

Justify your answer.

7. (10%) There are some bugs in the following code of the partition function in **quicksort**, please correct them.

```
private static int partition(Comparable[] a, int lo, int hi)
{
    int i = lo, j = hi;
    while (true)
    {
        while (less(a[++i], a[lo]))
            if (i == hi) break;
        while (less(a[lo], a[--j]))
            if (j == lo) break;
        exch(a, i, j);
    }
    exch(a, lo, i);
    return i;
}
```

8. (10%) **Write a recursive function** that reads in a sequence of characters (stored in a char array), and determines whether its parentheses, braces, and curly braces are "balanced." (Note: you cannot use **stack** in your code)

```
boolean balanced(char[] a) // e.g. return true for [()]{}{[()()]()}, false for [(])
```