CSE 332 : 22Su Final Pt. 1

Name:

NetID: @uw.edu

Instructions

- The allotted time is **60** minutes. Please do not turn the page until the staff says so.

- This is a closed-book and closed-notes exam. You are not permitted to access electronic devices.

- Read directions carefully, especially for problems that require you to show work or provide an explanation.

- We can only give partial credit for work that you've written down.

- Unless otherwise noted, every time we ask for an $O, \Omega,$ or Θ bound, it must be simplified and tight.

- For answers that involve bubbling ○ or □, make sure to fill in the shape completely: ● or ■.

- If you run out of room on a page, indicate that the answer continues on the back of that page. Try to avoid writing on the very edges of the pages: we scan your exams and edges often get cropped off.

- Make sure you also get a copy of the formula sheet.

Advice

- If you feel like you're stuck on a problem, you may want to skip it and come back at the end if you have time.

- Look at the question titles on the cover page to see if you want to start somewhere other than problem 1.

- Remember to take deep breaths.

Question	Max points
1. Amdahl's Law	5
2. Parallel Code	14
3. Parallel Prefix Sum	9
4. Concurrency Issues	13
5. Sorting	14
Total	**55**

1. Amdahl's Law [5 points]

(a) What *fraction of a program must be parallizeable* in order to get 5x speedup on 15 processors?

You must show your work for any credit. For full credit give your answer as a number or a simplified fraction (not a formula).

2. Parallel Code [14 points]

In Java, using the ForkJoin Framework, write code to solve the following problem:
• Input: An array of Strings (does not contain duplicates).
• Output: The **greatest number of vowels** in a String and **its location** (index) in the Input array. In case of ties, the rightmost index is returned. If no Strings in the array contain any vowels, then return -1 as its index.

Examples:
• Input: ["bonjour", "hola", "howdy", "guten tag", "ciao"]
• Output: 3, 4

• Input: ["bnjr", "hl", "hwdy", "gtn tg", "c"]
• Output: 0, -1

• Input: ["bnjr", "hl", "hwdy", "gtn tg", "c", ""]
• Output: 0, -1

• Do **not** employ a sequential cut-off: **the base case should process one element.**
(You can assume the input array will contain at least one element.)
• Give a class definition, FindMax, **along with any other code or classes needed.**
• We have provided some of the code for you, you should also fill in the _____ part.

You may not use any global data structures or synchronization primitives (locks)

```java
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveAction;
import java.util.concurrent.RecursiveTask;

class Pair { // You are not required to use this class
    int numVowels;
    int index;
    public Pair (int numVowels, int index) {
        this.numVowels = numVowels;
        this.index = index;
    }
}

class Main {
    static final ForkJoinPool fjPool = new ForkJoinPool();

    _____ findMax (String[] array) {

        return fjPool.invoke(new FindMax(_____));
    }
}
```

Please fill in the two _____ above and write your code on the next page.
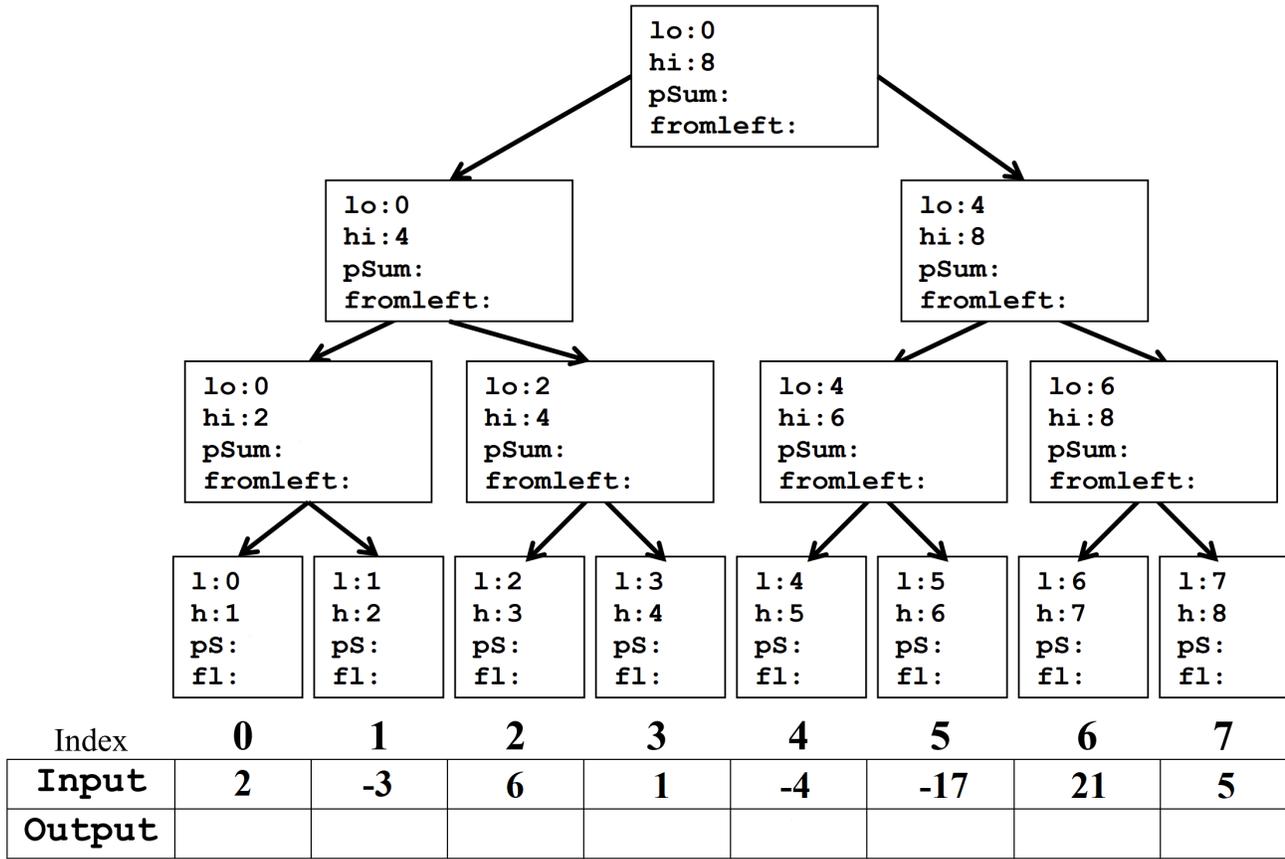
Write your class, FindMax, on this page, along with any other code or classes needed.
Please remember to also fill in the class header.

```
_____ {
    /*
     * Returns the number of vowels in the given String str
     */
    private int countNumVowels(String str) { // You are not required to use this method
        for (int i = 0 ; i < str.length(); i++){
            char ch = str.charAt(i);
            if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
                ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U'){
                count++;
            }
        }
        return count;
    }
}
```

3. Parallel Prefix Sum [9 points]

(a) (4 points) Dara has been training all year to climb Mt. Everest and she needs your help calculating if she will have enough energy to do so! In this particular problem, positive numbers will give Dara an energy boost and negative numbers will have no effect. Given the following array as input, perform the parallel prefix algorithm to fill the output array with the sum of **only the positive values contained in all of the cells to the left** (including the value contained in that cell) in the input array. Negative values in the input array should not contribute to the sum. Fill in the values for: **pSum, fromLeft**, and the **Output Array** in the tree and output array. Do not use a sequential cutoff.

```
                              lo:0
                              hi:8
                              pSum:
                              fromleft:

              lo:0                              lo:4
              hi:4                              hi:8
              pSum:                             pSum:
              fromleft:                         fromleft:

      lo:0          lo:2              lo:4          lo:6
      hi:2          hi:4              hi:6          hi:8
      pSum:         pSum:             pSum:         pSum:
      fromleft:     fromleft:         fromleft:     fromleft:

   l:0    l:1    l:2    l:3      l:4    l:5    l:6    l:7
   h:1    h:2    h:3    h:4      h:5    h:6    h:7    h:8
   pS:    pS:    pS:    pS:      pS:    pS:    pS:    pS:
   fl:    fl:    fl:    fl:      fl:    fl:    fl:    fl:
```

Index	0	1	2	3	4	5	6	7
Input	2	-3	6	1	-4	-17	21	5
Output								

(b) (3 points) Give formulas for the following values where **p** is a reference to a tree node other than a leaf node and **leaves[i]** refers to the leaf node in the tree visible just above the corresponding location in the **input** and **output** arrays in the picture above.

p.left.fromLeft=

p.right.fromLeft=

output[i]= leaves[i].psum + leaves[i].fromLeft

(c) (2 points) Describe how you assigned a value to **leaves[i].pSum**.

4. Concurrency [13 points]

After finishing up all of the grading for summer quarter, the Allen School TAs decide to celebrate by eating at their favorite restaurant on the Ave! The Restaurant class manages a restaurant's seating capacity through reservations and walk-ins. Multiple threads (TAs) could be accessing the same Restaurant object, which means two TAs could be making reservations or walk-ins at the same time. Assume the Queue objects ARE THREAD-SAFE, have enough space, and operations on them will not throw an exception.

```java
public class Restaurant {

    private Queue<Integer> reservations = new Queue<Integer>();
    private Queue<Integer> walkIns = new Queue<Integer>();
    private final int maxCapacity = 50; //Max capacity of people in restaurant

    // Calculates current restaurant capacity
    public int getTotal() {
        int total = 0;

        for (Integer group : reservations) {
            total += group;
        }

        for (Integer group : walkIns) {
            total += group;
        }

        return total;
    }

    // Adds list of reservation groups to the restaurant
    public void addReservations(List<Integer> groups) {
        for (int i = 0; i < groups.size(); i++) {
            if (getTotal() + groups.get(i) <= maxCapacity) {
                reservations.push(groups.get(i));
            }
        }
    }

    // Adds walkIn group to the restaurant
    public void addWalkIn(int walkInSize) {
        if (getTotal() + walkInSize <= maxCapacity) {
            walkIns.push(groupSize);
        }
    }

}
```

(a) (4 points) Neel and Thien (think of them as two threads) are both trying to access the same Restaurant object at the same time, but this potentially could cause some issues. If there are any problems, give an example of when and where they could occur. Be specific.

Does the `Restaurant` class as shown above have (bubble all that apply):

☐ a race condition   ☐ potential for deadlock   ☐ a data race   ☐ none of these

(b) (4 points) Hans proposes that we make the `addWalkIn` method synchronized in order to potentially eliminate any concurrency issues in the code, and change nothing else.

Does this modified `Restaurant` class have (bubble all that apply):

☐ a race condition   ☐ potential for deadlock   ☐ a data race   ☐ none of these

If there are any FIXED problems, describe why they are FIXED. If there are any NEW problems, give an example of when those problems could occur. Be specific!

(c) (5 points) Modify the code on the **next page** to use locks to allow the most concurrent access and to avoid all of the potential problems listed above. For full credit you must allow the most concurrent access possible without introducing any errors. Create locks as needed. Use any reasonable names for the locking methods you call. DO NOT use synchronized. You should create re-entrant lock objects as follows:

```
ReentrantLock lock = new ReentrantLock();
lock.acquire();
lock.release();
```

Answer this question in the next page.

```java
// fix this code with locks!
public class Restaurant {

    private Queue<Integer> reservations = new Queue<Integer>();
    private Queue<Integer> walkIns = new Queue<Integer>();
    private final int maxCapacity = 50; //Max capacity of people in restaurant



    // Calculates current restaurant capacity
    private int getTotal() {

        int total = 0;

        for (Integer group : reservations) {
            total += group;
        }

        for (Integer group : walkIns) {
            total += group;
        }

        return total;
    }

    // Adds list of reservation groups to the restaurant
    public void addReservations(List<Integer> groups) {

        for (int i = 0; i < groups.size(); i++) {

            if (getTotal() + groups.get(i) <= maxCapacity) {

                reservations.push(groups.get(i));

            }

        }

    }

    // Adds walkIn group to the restaurant
    public void addWalkIn(int walkInSize) {

        if (getTotal() + walkInSize <= maxCapacity) {

            walkIns.push(groupSize);

        }

    }

}
```

8

5. Sorting [14 points]

(a) (3 points) Perform **one pass** of Hoare's partition on a following data. We already swapped the pivot, 35, to the first cell for you (the first to the second row). Show every swap, one per row. Don't forget to swap the pivot back.

3	14	15	92	65	35	89	79	32	38	46

35	14	15	92	65	3	89	79	32	38	46

(b) (4 points) Perform a radix sort on the following data.

23	14	6	92	63	27	79	26	90	5

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

(c) (4 points) Nathan wants to create a new data structure for priority queue operation, SuperQueue, with run-time of $\Theta(1)$ for insert() and $\mathcal{O}(\log \log n)$ for removeMin(). Do you think this is possible? Explain briefly. (**Hint:** This is a sorting question.)

(d) (3 points) Give the **recurrence** for the PARALLEL MERGE discussed in lecture — **worst case span**. Note: We are NOT asking for the closed form. This is just a merge routine, *not full Mergesort*. **For any credit, explain all parts of your answer briefly.**

$$T(N) =$$

Extra piece of paper for scratch work

Reference Sheet

Geometric series identities

$$\sum_{i=0}^{k} c^i = \frac{c^{k+1} - 1}{c - 1} \qquad \sum_{i=0}^{\infty} c^i = \frac{1}{1 - c} \text{ if } |c| < 1$$

Sums of polynomials

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2} \qquad \sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \qquad \sum_{i=0}^{n} i^3 = \frac{n^2(n+1)^2}{4}$$

Log identities

$$b^{\log_b(a)} = a \qquad \log_b(x^y) = y \cdot \log_b(x) \qquad a^{\log_b(c)} = c^{\log_b(a)} \qquad \log_b(a) = \frac{\log_d(a)}{\log_d(b)}$$

Exponent properties

$$(a^m)^n = a^{m \cdot n} = (a^n)^m$$