Name:____

CSE373 Winter 2014, Final Examination
March 18, 2014

Please do not turn the page until the bell rings.

Rules:

- The exam is closed-book, closed-note, closed calculator, closed electronics.
- **Please stop promptly at 4:20.**
- There are **98 points** total, distributed **unevenly** among **10** questions (many with multiple parts):

Question	Max	Earned
1	7	
2	7	
3	8	
4	20	
5	8	
6	8	
7	16	
8	8	
9	7	
10	9	

Advice:

- Read questions carefully. Understand a question before you start writing.
- **Write down thoughts and intermediate steps so you can get partial credit. But clearly circle your final answer.**
- The questions are not necessarily in order of difficulty. **Skip around.** Make sure you get to all the problems.
- If you have questions, ask.
- Relax. You are here to learn.

Name:____

1. (**7** points)

   Given this code, write client code using `BankAccount` such that a call to `getOwnerAge` encounters a `NullPointerException`. Then explain briefly how to rewrite one of the methods in `BankAccount` to avoid this problem.

```
class Person {
    public String name;
    public Date birthdate; // has a Date(int year, int month, int day) constructor
}

class BankAccount {
    private Person owner;
    private float balance;
    public BankAccount(Person o, float b) {
        if (o == null || o.birthdate == null) {
            throw new IllegalArgumentException();
        }
        owner = o; balance = b;
    }

    public long getOwnerAge() {
        Date now = new Date(); // initialized to be the current date
        long millisecondsPerYear = 365 * 24 * 60 * 60 * 1000;
        // the Date getTime() method returns the date as the number
        // of milliseconds since January 1, 1970, 00:00:00 GMT
        return (now.getTime() - owner.birthdate.getTime()) / millisecondsPerYear;
    }
}
```

Name:____

2. (**7** points)

   Your friend wrote the code below to compute the maximum value from an array of integers. Unfortunately, your friend is not seeing the speedup they were expecting. Briefly explain why this is the case (i.e. identify the mistake they made). Then modify their code to take full advantage of the available parallelism. Indicate which lines of the original you are replacing.

```
class MaxThread extends java.lang.Thread {
    int lo; int hi; int[] arr; // arguments
    int ans = Integer.MIN_VALUE; // result
    MaxThread(int[] a, int l, int h) {  }
    public void run() { // override
        if(hi - lo < SEQUENTIAL_CUTOFF)
            for(int i=lo; i < hi; i++)
                if (arr[i] > ans)
                    ans = arr[i];
        else {
            MaxThread left = new MaxThread(arr,lo,(hi+lo)/2);
            MaxThread right= new MaxThread(arr,(hi+lo)/2,hi);
            left.run();
            right.run();
            ans = Math.max(left.ans, right.ans);
        }
    }
}

int max(int[] arr){
    MaxThread t = new MaxThread(arr,0,arr.length);
    t.run();
    return t.ans;
}
```

Name:____

3. (**8** points)

   You are at a summer internship working on a program that currently takes 12 minutes to run on a 2-processor machine. Two-thirds of the execution time (8 minutes) is spent running sequential code on one processor and the other third is spent running parallel code on both processors. Assume the parallel code enjoys perfectly linear speedup for any number of processors.

   Note/hint/warning: This does *not* mean two-thirds of the work is sequential. Two-thirds of the *running time* is spent on sequential code.

   Your manager has a budget of $2,000 to speed up the program. She figures that is enough money to do only one of the following:

   - Buy a 8-processor machine.
   - Hire a CSE373 graduate to parallelize more of the program under the highly dubious assumptions that:
     - Doing so introduces no additional overhead
     - Afterwards, there will be 1 minute of sequential work to do and the rest will enjoy perfect linear speedup.

   Which approach will produce a faster program? Show your calculations, including the total *work* done by the program and the expected running time for both approaches.

Name:____

4. (**20** points)

   Below are five computational tasks. For each one, **choose the data structure or abstract data type** you feel best suited to the task from the following list: Stack, Queue, Hashtable, AVL Tree, Priority Queue, Union-Find. In addition, **list the basic operations** for the data structure or ADT that you chose and **give asymptotic worst-case running times for those operations**. No additional explanation is required.

   (a) While processing a list of objects, check if you have processed a particular object before.

   (b) Store a list of students and their grades. You must also provide an efficient way for a client to see all students sorted in alphabetical order by name. Give the running time for this operation as well.

   (c) Process a digital image to divide the image up into groups of pixels of the same color.

   (d) Compute a frequency analysis on a file. That is, count the number of times each character occurs in the file, and store the results.

   (e) Store the activation records (i.e. objects containing the return address and local variable associated with a function call) for nested function calls.

Name:____

5. (**8** points)

   Suppose we use radix sort to sort the numbers below, using a radix of 10. Show the state of the sort *after each of the first two passes*, *not* after the sorting is complete. If multiple numbers are in a bucket, put the numbers that "come first" closer to the top.

   Numbers to sort (in their initial order):

   17, 45, 877, 31, 7, 222, 42, 43, 301, 2525, 9, 27, 64

   Put the result after the first pass here:

0	1	2	3	4	5	6	7	8	9

   Put the result after the second pass here:

0	1	2	3	4	5	6	7	8	9

Name:____

6. (**8** points)

   For each of the following situations, name the best sorting algorithm from among those we studied. There may be more than one answer deserving full credit, but you only need to give one answer for each.

   (a) You need a very fast sort on average, and you can only use a constant amount of extra space.

   (b) The array is in perfect sorted order.

   (c) You have a large data set, but you know all the values are between 0 and 999.

   (d) Copying your data is very fast, but comparisons are relatively slow.

   Please provide an example of each of the following (or say NONE if no example exists) from among the sorting algorithms we studied.

   (e) A stable comparison sort with a worst-case $O(n^2)$ running time

   (f) A stable comparison sort with a worst-case $O(n)$ running time

   (g) An efficient (i.e. $O(n \log n)$) stable comparison sort

   (h) A stable sort with a worst-case running time linear in n

Name:____

7. (**16** points)

Below are four graph-based computational tasks. For each one, **specify the type of graph** most appropriate for the data in question in terms of undirected or directed, and unweighted or weighted. In addition, **choose the graph algorithm** from the following list best suited to computing a solution: Breadth-First Search, Minimum Spanning Tree (e.g. Prim's or Kruskal's), Dijkstra's Algorithm, Topological Sort, Depth-First Search. No additional explanation is required.

(a) You have data for the rail transport (i.e. train track) network in North America, including the length of each section of track. You need to compute the shortest (in terms of distance) route for a train traveling from New York City to Atlanta, GA.

(b) You need to lay water lines for a new housing development (i.e. pipes to carry water to and from each house). You have data on all the places it's possible to build pipes, and how much pipe would be required in each case. From the many potential pipeline options, you must compute which pipes to actually build such that you use the least amount of pipe overall.

(c) You are compiling a program from source code and you know the dependencies between source files (i.e. for each file F, you know which other files, if any, can't be compiled until F is compiled). You need to compute a valid compilation order such that each file is compiled after all of its dependencies are compiled.

(d) You have data about friend relationships from a social network, and you want to model the spread of a rumor based on these relationships. Your model is that a rumor starts with a single person, who then tells all of their friends. Then at each subsequent step, everyone who knows about the rumor spreads it to all of their friends who don't know it. You want to compute how many people know about the rumor after k steps.

Name:____

8. (**8** points)

   (a) Why might it be faster to access all the elements of an array-based queue than to access all the elements of a linked-list-based queue?

      i. temporal locality
      ii. spatial locality
      iii. it is faster asymptotically
      iv. none of the above

   (b) Why might the second version of `f` be faster than the first?

```
public void f(String[] strings) {
    for (int i = 0; i < strings.length; i++) {
        strings[i] = strings[i].trim(); // omits trailing/leading whitespace
    }
    for (int i = 0; i < strings.length; i++) {
        strings[i] = strings[i].toUpperCase();
    }
}

public void f(String[] strings) {
    for (int i = 0; i < strings.length; i++) {
        strings[i] = strings[i].trim(); // omits trailing/leading whitespace
        strings[i] = strings[i].toUpperCase();
    }
}
```

      i. temporal locality
      ii. spatial locality
      iii. it is faster asymptotically
      iv. none of the above

   (c) In any graph, why might checking if an edge exists be faster when using an adjacency matrix to represent the graph instead of an adjacency list?

      i. temporal locality
      ii. spatial locality
      iii. it is faster asymptotically
      iv. none of the above

   (d) In a dense graph, why might finding all the vertices adjacent to a particular vertex be faster when using an adjacency matrix to represent the graph instead of an adjacency list?

      i. temporal locality
      ii. spatial locality
      iii. it is faster asymptotically
      iv. none of the above

Name:____

9. (**7** points)

   (a) A *data race* is situation where the behavior of an application depends on the sequence or timing of parallel threads. That is, the application behavior may depend on which thread ”gets there first.” Which of the Java `Thread` class methods can be used to help prevent data races?

      i. `run`
      ii. `join`
      iii. `hashCode`
      iv. `start`

   (b) For ***each*** of the following, indicate whether it can be computed with:

      - A parallel map operation
      - A parallel reduce operation
      - Neither

      i. Increment each integer in an array.
      ii. Compute the mode (i.e. the most frequently occurring value) of an array of integers.
      iii. Convert an array of strings to an array of integers by parsing each string as an integer.
      iv. Compute the average of an array of numbers.
      v. Find the minimum element in an array of numbers.

Name:____

10. (**9** points)

(a) Which of the following would generally be the best choice for computing the hashcode of an object with multiple fields?

i. Use the product of the hashcodes for each field.

ii. Take the maximum of the hashcodes for all the fields and multiply it by a large prime number.

iii. Sum the hashcodes for all the fields, multiplying by a prime number before each addition.

iv. Randomly choose a field and use its hashcode.

(b) Java expects a particular relationship between the equals and hashCode methods of any object. What is that relationship?

(c) Briefly describe a good approach for generating a hashcode from a string.

(d) Why is quicksort not a good choice for an external sorting algorithm given data stored on traditional hard drives?

(e) Suppose we can randomize the order of an array of elements in $O(n)$ time. For which sorting algorithms (if any) could it be worthwhile to randomize the input array before performing the sort?

Name:____

An extra page in case you find it useful