CS 62 Practice Final

2017-5-2

Please put your name on the back of the last page of the test.

Note: This practice test may be a bit shorter than the actual exam.

Part 1: Short Answer [32 points]

1. [4 points] In an array-based representation of a tree, **what's the formula** for the index of the child of the node at index i? **Are the left and right children** of a node always adjacent to each other in the array, or not?

2. [4 points] **Explain the difference** between weak and strong connectivity in a directed graph.

3. [4 points] **Explain the practical purpose** of a sequential cutoff in code that uses the map-reduce paradigm (like the `ParallelSum` example we saw in class).

4. [4 points] **What's the difference** between a callback and a normal function in terms of when they are called?

5. [4 points] **Explain the problem** with the following code, and suggest how to fix it:

```
public void printEven(Iterator<Integer> iter) {
  while (iter.hasNext()) {
    if (iter.next() % 2 == 0) {
      System.out.println(iter.next());
    }
  }
}
```

Hint: it isn't using the ***`next`*** *function correctly.*

6. [4 points] If a program takes 30 seconds to set up some parallelizable work and then another 30 seconds to do that work sequentially, **what is the maximum achievable speedup** for this program using parallelism according to Amdahl's law?

7. [4 points] **Explain what** the `malloc` function does in C. **Also explain** what its argument does.

8. [4 points] In a graph where the shortest path from A to B is 3 edges, while the shortest path from A to C is 7 edges, **will breadth-first search** starting at A explore B or C first, or does it depend on the structure of the graph? **Justify your answer.**

Problem 2: Linked Lists [12 points]

1. **Diagram** [4 points]

Draw a diagram of a doubly-linked list with both head and tail pointers that contains the elements 17, 23, and 31. **Use arrows to indicate** pointers, including the head and tail pointers, and the next and previous pointers of each node. **For null pointers**, draw an arrow pointing to the word “NULL.”

2. **Operations** [4 points]

Assuming we wanted to insert the value 47 between the 23 and 31 in our list, **write a list of which pointers** would have to be changed. **Include pointers** that are part of the new node.

3. **Efficiency** [4 points]

If you use this doubly-linked list to implement a queue, **what would** the big-O run times of the push and pop methods be? Also, **what would be the run time** of an “insertSorted” method that inserted an item such that the whole list remained in sorted order?

Problem 3: Malloc and Free [12 points]

1. **Malloc** [6 points]

For the code fragment below, **how many different** heap regions does it allocate, and **how many bytes** does it allocate on the heap. Also, when the setup function is called, **how many bytes** are allocated on the stack for its local variable(s)? *Assume the size of an int is 4 bytes, and the size of a pointer of any type is 8 bytes.*

```
int** setup() {
    int** arrays = malloc(2 * sizeof(int*));
    arrays[0] = malloc(4 * sizeof(int));
    arrays[1] = malloc(4 * sizeof(int));
    return arrays;
}
```

2. **Free** [6 points]

Fill in the cleanup function below so that when given a pointer returned by the setup function from part 1, it will properly free all of the allocated memory.

```
void cleanup(int** arrays) {




}
```

Problem 4: Graphs [12 points]

1. **Acyclic Non-Trees** [4 points]

Draw a directed, acyclic graph with 5 nodes which is **not** a tree, but which has at least one spanning tree. *Label your nodes with the letters A through E.*

2. **Spanning Trees** [4 points]

How many spanning trees does the graph that you drew in part 1 have? **List the root nodes** of each spanning tree (list the same node multiple times if it's the root of multiple spanning trees).

3. **Paths** [4 points]

Is the graph you drew in part one **strongly connected, weakly connected, both, or neither**? **Write down a sequence of nodes** which is the longest (or a tied-for-longest) path in that graph.

Name: ____________________