CPSC 320 Sample Final
April, 2016

Name: ______________________ Student ID: ______________________

Signature: ______________________

– You have 2.5 hours to write the 9 questions on this examination. A total of 140 marks are available.

– **Justify all of your answers.**

– You are allowed to bring in three hand-written or printed, double-sided 8.5x11in sheet of notes, and nothing else.

– Keep your answers short. If you run out of space for a question, you have written too much.

– The number in square brackets to the left of the question number indicates the number of marks allocated for that question. Use these to help you determine how much time you should spend on each question.

– Use the back of the pages for your rough work.

– **Good luck!**

Question	Points	Score
1	24	
2	20	
3	20	
4	8	
5	12	
6	10	
7	14	
8	16	
9	16	
Total:	140	

1. [24 points] Short Answers

(a) [6 points] Consider the Gale-Shapley stable matching algorithm. Show that at any step of this algorithm the following claim is **true**: *if there is a free man, then there is a woman he has not proposed to yet*.

(b) [4 points] Write down the exact (mathematical) definition of what it means that $f(n) \in O(g(n))$.

(c) [8 points] In each row below, circle the correct statement (either (a) or (b) or (c)) if we know that for every $n \in \mathbb{N}$, $0 < f(n) < g(n)$:

(a) $f(n) \in O(g(n))$ (b) $f(n) \notin O(g(n))$ (c) $f(n)$ may or may not be in $O(g(n))$

(a) $f(n) \in \Omega(g(n))$ (b) $f(n) \notin \Omega(g(n))$ (c) $f(n)$ may or may not be in $\Omega(g(n))$

(a) $f(n) \in o(g(n))$ (b) $f(n) \notin o(g(n))$ (c) $f(n)$ may or may not be in $o(g(n))$

(a) $f(n) \in \omega(g(n))$ (b) $f(n) \notin \omega(g(n))$ (c) $f(n)$ may or may not be in $\omega(g(n))$

(d) [6 points] What are the two strategies used to show that the greedy solution is an optimal solution? Briefly explain what is the main idea of each strategy.

2. [20 points] *Short answers.*

   (a) [10 points] The worst-case running time of an algorithm you wrote satisfies the recurrence relation

$$T(n) = \begin{cases} aT(n/4) + n^x & \text{if } n \geq 4 \\ \Theta(1) & \text{if } n < 4 \end{cases}$$

   For which values of the constants a and x will the algorithm run in $\Theta(n^2)$ time?
   Hint: there are two cases.

   (b) [4 points] In our divide and conquer algorithm to find the closest pair of points in the plane, why was it sufficient, during the merge step, to consider points that were at most 11 positions apart in the list of points in the vertical strip around the dividing line sorted by their y-coordinates?

(c) [6 points] Assume n is even. Describe an algorithm that find simultaneously the minimum and the maximum among n elements using $3n/2 - 2$ comparisons in the worst case.

3. [20 points] You and a group of your friends want to play tug-of-war (two teams pulling on opposites sides of a rope). Being the only computer scientist in the group, you have been asked to design an algorithm to build two teams A and B that are as equal as possible. You formalize the problem as follows:

   - Each person i has a strength s_i.
   - You would like the sum of the strengths of the people on team A to be as close to equal as possible to the sum of the strengths of the people on team B. That is, you want to minimize

$$\left| \sum_{i \in A} s_i - \sum_{j \in B} s_j \right|$$

   Note that the two teams do not need to have the same number of people.

   (a) [12 points] Write a pseudocode for a **greedy** algorithm to build the teams A and B. Your algorithm does not need to always succeed at minimizing the difference in total strength between the two teams, but it should make a good attempt at it.

(b) [4 points] Analyze the time complexity of your algorithm from part (a). Specify only as much of your implementation (for instance, data structures) as needed for your analysis. Try to obtain as tight bound on the running time as possible, but make sure that your answer is correct (otherwise you will lose all points for this part). Justify your answer!

(c) [4 points] Give an example where your algorithm will not return the optimal solution (the one that minimizes the strength difference between the two teams).

4. [8 points] Let $G = (V, E)$ be an undirected graph with costs $c(e) \geq 0$ for all edges $e \in E$. Assume that the edge costs are distinct. Assume you are given a minimum-cost spanning tree T in G. (You don't need to find or verify it, you can assume it is a minimum-cost spanning tree). Now assume that a new edge is added to E, connecting two nodes $v, w \in V$ with cost c that is distinct from costs of edges already in the graph.

   Describe an efficient algorithm to test if T remains the minimum-cost spanning tree with the new edge added to G (but not to the tree T). Make your algorithm run in time $O(|E|)$ time. Please note any assumptions you make about what data structure is used to represent the tree T and the graph G. Justify briefly why your algorithm is correct.

5. [12 points] Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some integer d. Each node v of T is labeled with a real number x_v. You may assume that the real numbers labeling the nodes are all distinct. A node v of T is a **local minimum** if the label x_v is less than the label x_w for all nodes w that are connected to v by an edge.

   You are given such a complete binary tree T, but the labeling is only specified in the following way: for each node v, you can determine the value x_v by **probing** the node v. **Design** a **divide and conquer** recursive algorithm that find a local minimum in a T and show that the number of probes it will make is in $O(\log n)$. Write *pseudocode* to specify your algorithm. Remember to justify that your algorithm will make $O(\log n)$ probes.

6. [10 points] Consider a marking caching algorithm from class:

```
1:  procedure MarkingAlgorithm($\sigma$)
2:      start with all cache items unmarked
3:      for a request to item $s$ in $\sigma$ do
4:          if $s$ is not in the cache then
5:              if all cache items are marked then
6:                  unmark all cache items                    ▷ new phase
7:              end if
8:              evict an unmarked item from the cache
9:          end if
10:         mark $s$
11:     end for
12: end procedure
```

Let r be the number of phases and let c_j be the number of fresh items in phase j (not requested in the previous phase or not initially in the cache if $j = 1$).

Second, consider the optimal (greedy) algorithm from class (furthest-in-future). Let f_j be the number of evictions/misses the algorithm does in phase j.

(a) [2 points] Argue that $f_1 \geq c_1$.

(b) [4 points] Argue that for every $j < r$, $f_j + f_{j+1} \geq c_{j+1}$.

(c) [4 points] Show that the number of evictions/misses of the optimal greedy algorithm is at least $\frac{1}{2}\sum_{j=1}^{r} c_j$.

7. [14 points] Suppose we have a binary counter. The counter is represented by a singly linked list of bits, where the least-significant (rightmost) bit is at the head of the list. For instance, a counter containing the value 10010 would be represented by the linked list $head \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow 1 \rightarrow nil$. The counter is initialized at 0 (so the linked list is initially $head \rightarrow 0 \rightarrow nil$). The counter supports the following two operations:

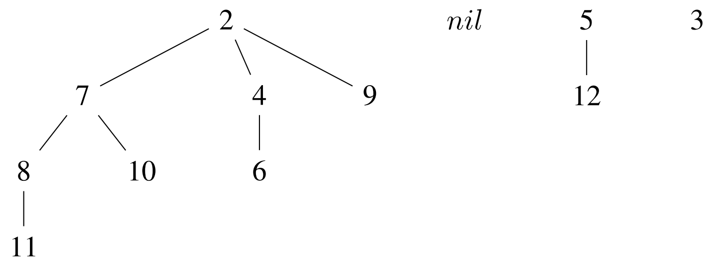
   - INCREMENT: adds 1 to the counter.
   - DOUBLE: doubles the value of the counter.

   Use amortized analysis to show that the worst-case running time of any sequence of n INCREMENT and DOUBLE operations is in $O(n)$.

8. [16 points] Binomial Heaps.

   (a) [4 points] Explain why a binomial tree of order k has 2^k nodes.

(b) [6 points] Consider the following state of the binomial heap (nil represents the empty element in the ordered list):



Draw the state of the binomial heap after operation ExtractMin(). Circle your final answer.

(c) [2 points] Assume that a binary heap contains 36 elements. How many binomial trees does it contain and what are their orders.

(d) [4 points] Discuss the relationship between inserting an element into a binomial heap and incrementing a binary counter by one. Explain why this implies that building a binomial heap takes $O(n)$ time.

9. [16 points] **Knapsack Problem**: Given n items, where the i-th item has weights w_i and value v_i, and given value W, the goal is to find a subset S of the items with **the maximum sum of their values** subject to the **restriction** that the total weight of chosen items is **at most** W. Assume that W and $w_1, \ldots, w_n$ are positive integers.

   Define the subproblems and specify the **recurrence relation** between the optimal values of subproblems.

   Then write a **pseudocode** for a dynamic programming algorithm with complexity $O(nW)$ that finds the value of an optimal solution (it does not need to find an optimal solution). **Do not use** the memoization technique: your algorithm should be iterative, not recursive.

   Finally, write a pseudocode for a backtracking algortihm that would reconstruct an optimal solution.

End of exam.