PSU CS 584/684 Final Spring 2014 Name:_____

Before you start, ***write your name at the top of each page****. You may use space on the front of this paper, and you may add extra pages as needed. Put your name on these as well.*

I suggest that you read the entire exam before you start. If you find a problem with the exam, please note it in your answer and answer as best you can. Please show as much of your work as you reasonably can.

Please answer all questions to the best of your ability. Allow yourself four hours of desk time for this exam: you may take breaks as needed. You may use your textbook and notes, but please do not consult other resources.

1. Several years ago I helped a company, SA, that had an interesting problem with research in grocery stores. SA had a collection C of possible shopping cart contents: each item $c \in C$ had two parts: a guess $c.p$ at a shopper's purchases based on observation during the shopping trip and an approximate timestamp $c.t$ (in seconds) at which the shopper reached the register. SA also had a collection R of register receipts with each $r \in R$ containing a checkout timestamp $r.t$ (in seconds) at which the goods were purchased.

   Unfortunately, because of errors in observation, SA needed to approximately match cart contents to receipts. To make this work, they constructed a scoring function $f : C \times R \rightarrow \{0..1\}$ that gave a heuristic probability that a given cart and receipt matched. Further, they decided to only try to match a cart c with a receipt r if $|c.t - r.t| < 300$.

   You are to construct a reasonably efficient algorithm that does a reasonable job of matching carts with receipts. That is, given C, R (not necessarily of equal size, since things might be missing) and f, you are to produce a matching of carts to receipts such that each cart in the matching is paired with exactly one receipt, and that gives a reasonably high value for the product of f over all elements of the matching.

   Start by formalizing the problem you are trying to solve, then give pseudocode for an algorithm, then give an asymptotic complexity analysis of your algorithm. Finally, comment on what performance you actually expect when given an instance with around 10,000 carts and receipts. Be sure to mention any assumptions you have made in formalizing and solving the problem.

2. The following problem is NP-complete:

Number 2-Partitioning

Instance: A collection C of positive integers, with $\sum_{c \in C} c$ even.

Find: A partition of C into two collections C_1 and C_2 that have equal sum; that is:

$$\sum_{c \in C_1} c = \sum_{c \in C_2} c$$

Use this fact to show that the following problem is also NP-complete:

Number 3-Partitioning

Instance: A collection C of positive integers, with $\sum_{c \in C} c$ evenly divisible by 3.

Find: A partition of C into three collections C_1, C_2, C_3 that have equal sum; that is:

$$\sum_{c \in C_1} c = \sum_{c \in C_2} c = \sum_{c \in C_3} c$$

3. As discussed in class, a lazy queue can be implemented using two stacks. One stack is used for enqueueing, and the other for dequeuing. When the dequeueing stack is empty, the enqueing stack is popped onto it one element at a time. The fundamental operations are:

   To create a new queue:
   $q.in \leftarrow$ new empty stack
   $q.out \leftarrow$ new empty stack
   return q

   To enqueue x in q:
   push x onto $q.in$

   To dequeue a value from q:
   if $q.out$ is empty
   while $q.in$ is not empty
   $v \leftarrow$ pop $q.in$
   push v onto $q.out$
   if $q.out$ is empty
   fail
   $v \leftarrow$ pop $q.out$
   return v

   Assume that stack operations are $O(1)$. Show that the amortized complexity of queue operations is also $O(1)$ for any legal sequence of queue operations.