ALGORITHMS FINAL

1. (60 pts.) Imagine a thief entering a house. In the house, there are infinitely many items that can have only one of three different weights: *1* kg, *3* kgs, and *5* kgs. All of the items are discrete. The thief has a bag capacity of *n* kgs and strangely, he wants to steal the "smallest number of items".

- **(a)** (10 pts.) Give a mathematical recursive formulation for *C(n)* where *C(n)* denotes the smallest number of items the thief can steal using a bag capacity of *n*.
- **(b)** (5 pts.) Show that this problem has the overlapping subproblems property.
- **(c)** (15 pts.) Write a recursive algorithm (as a pseudocode) that returns the smallest number of items the thief can steal using a bag capacity of *n*.
- **(d)** (5 pts.) Show that the greedy choice of taking the largest weight items into the bag first fails to lead to an optimal solution.
- **(e)** (15 pts.) Write a dynamic programming algorithm (as a pseudocode) for finding the smallest number of items the thief can steal using a bag capacity of *n*.
- **(f)** (10 pts.) Provide the running time of your dynamic programming algorithm. Explain.

2. (20 pts.) Assume that you are creating an array data structure that has a fixed size of *n*. You want to backup and empty this array after every *n* insertion operations. Unfortunately, the backup operation is quite expensive, it takes *n* time to do the backup. Insertions without a backup just take *1* time unit. Show that you can do backups in *O(1)* amortized time.

- **(a)** (10 pts.) Use the accounting method for your proof. Explain in sufficient detail.
- **(b)** (10 pts.) Use the potential method for your proof. Explain in sufficient detail.

3. (20 pts.) A university has two student clubs. The number of students registered to the first club is *m* and their IDs are stored in an array *A* (with *m* elements) whereas the number of students registered to the second club is *n* and their IDs are stored in an array *B* (with *n* elements), where $m \leq n$. A student might be registered to either one of these clubs or both. We want to decide how many students are registered to both clubs.
Given two arrays *A* and *B* along with their lenghts *m* and *n*, write a *O(mlogn)* algorithm (as a pseudocode) to find the number of elements that are registered to both clubs. For example, when *A* is *[2, 6, 3, 9, 11, 8]* and *B* is *[3, 11, 7, 4, 2, 5, 1]*, the algorithm must return *3* corresponding to the students with IDs *2, 3* and *11*. Inside your pseudocode, you are allowed to use functions that are already defined in class videos, slides and book. Also, explain why your running time is *O(mlogn)* in sufficient detail.