# CSE 332 Data Abstractions, Winter 2015 Homework 4 – Part A

Due: Wednesday, Feb 4, 2015 at 23:00 (11:00pm) via the catalyst drop box. You should refer to the written homework guidelines on the course website for a reminder about what is acceptable pseudocode. Part A of this homework has TWO terrific questions!! Your combined score from part A & part B will be your total HW4 score.

#### Submission instructions

Submit an electronic copy to the catalyst dropbox as a PDF file. You can either do the assignment on an electronic word processor (and convert to PDF) or do it on physical paper and scan it (or take a high res photo) and upload a single PDF of the file. It will be much easier to grade if <u>every question starts on a separate page</u>. Don't forget to put your name on the top of the first page.

#### Problem 1: Deletion in Hashing

In this problem you will think about how lazy deletion is handled in open addressing hash tables. (a) Suppose a hash table is accessed by open addressing and contains a cell X marked as "deleted". Suppose that the next successful find hits and moves past cell X and finds the key in cell Y. Suppose we move the found key to cell X, mark cell X as "active" and mark cell Y as "open". Suppose this policy is used for every find. Would you expect this to work better or worse compared to not modifying the table? Explain your answer.

(b) Suppose that **instead** of marking cell Y as "open" in the previous question, you mark it as "deleted" (it contains no value, but we treat it as a collision). Suppose this policy is used for every find. Would you expect this to work better or worse compared to **not modifying** the table? Explain your answer.

## (See back of this page for other problem)

### Problem 2: Algorithm Analysis

The methods below implement recursive algorithms that return the first index in an *unsorted* array to hold 17, or -1 if no such index exists.

```
int first17 a(int[] array, int i) {
                                            int first17 b(int[] array, int i) {
  if (i >= \overline{array}.length)
                                              if (i >= \overline{array.length})
      return -1;
                                                   return -1;
  if (array[i]==17)
                                               if (array[i]==17)
     return 0;
                                                  return 0;
  if (first17 a(array,i+1) == -1)
                                              int x = first17 b(array,i+1);
      return -1;
                                               if (x == -1)
  return 1 + first17 a(array,i+1);
                                                   return -1;
                                              return x + 1;
}
                                             }
```

(a) What kind of input produces the worst-case running time in an absolute "number of operations" sense, not a big-O sense, for first17\_a (arr,0)?

(b) For first17\_a, give a recurrence relation, including a base case, describing the worst-case running time, where n is the length of the array (e.g. T(n) = ...). You may use whatever constants you wish for constant-time work.

(c) Give a tight asymptotic ("big-Oh") upper bound for the running time of first17\_a(arr,0) given your answer to the previous question. That is, find a closed form for your recurrence relation. Show how you got your answer.

(d) What kind of input produces the worst case running time in an absolute "number of operations" sense, not a big-O sense, for first17\_b(arr,0)?

(e) For first17\_b, give a recurrence relation, including a base case, describing the worst-case running time, where n is the length of the array. You may use whatever constants you wish for constant-time work.

(f) Give a tight asymptotic ("big-Oh") upper bound for the running time of first17\_b(arr,0) given your answer to the previous question. That is, find a closed form for your recurrence relation. Show how you got your answer.

(g) Give a tight asymptotic ("big-Omega") worst-case lower bound for the *problem* of finding the first 17 in an unsorted array (not a specific algorithm). Briefly justify your answer.