# CS 211 Homework #3

Please complete the homework problems on the following page using a separate piece of paper. Note that this is an individual assignment and all work must be your own. Be sure to show your work when appropriate. This assignment is due **in class** on Tuesday, November 29, 2016.

1. In this question, you will be evaluating the efficiency of various ways to handle hash table collision. Assume that for integers, our hash table uses the following hash function:

hash(x) = (2 \* (x^3) + 7) % <table size>

In this exercise, you will evaluate hash tables that use the following collisions rules:

1. Separate chaining (i.e. buckets)
2. Linear probing whose probe for the next space = (i + 1)%<table size>. E.g. If the hash method computes location 0, the next probe would be at (0 + 1) = 1.
3. Quadratic probing whose probe for the next space = (i^2 + 1) % <table size>.
4. Double hash probing. We didn't talk about this method in class, but the basic premise is that the next location is calculated by using a secondary hashing function. Let the secondary hashing function be:  
   hash2(x) = 1 + i \* (13 - (x % 7)). For example, given x = 2, i = 3 with a table size of 10, hash2(2) = (1 + 3 \* (13 - (1 % 7))) % 10 = 7.

Starting with an empty hash table with a fixed size of 11, insert the following keys in order into four distinct hash tables (one for each collision mechanism): {12, 31, 1, 0, 42, 98, 70, 32, 33}. You are only required to show the final result of each hash table. In the very likely event that a collision resolution mechanism is unable to successfully resolve, simply record the state of the last successful insert and note that collision resolution failed.

[3] Separate Chaining (buckets)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |

### [3] Linear Probing

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |

### [3] Quadratic Probing

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |

### [3] Double Hashing

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |

2.[5] **Radix Sort.** Perform a radix sort on the following list: {77, 98, 123, 256, 789, 101, 112, 131, 415, 164, 718, 292, 401, 699}. Use the table below to show your work.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

3. [5] **Merge Sort**. Perform a merge sort on the following list: {77, 98, 123, 256, 789, 101, 112, 131, 415, 164, 718, 292, 401, 699}. Show your answer in graphical form (i.e. pretty pictures).

4. [5] **Quick Sort**. Perform two iterations of quick sort on the following list: {77, 98, 123, 256, 789, 101, 112, 131, 415, 164, 718, 292, 401, 699}.