Due date: November 18 @ 11:55pm

Submission Instructions:

• Each person must submit a solution to this assignment on Moodle on the due date.

• Assignments must be typed. I suggest using Latex. If the grader (me!) cannot (easily) read an answer, she will give it a 0.

• All arguments should be clear, concise, and (of course) correct. You will lose points for poor writing, bad grammar, etc...

• Use of the Internet for solving these problems is strictly forbidden, and will be treated as an honor code violation.

• Working together: You may may discuss the questions with each other. However:
  – With your submission you must provide a list of all people you discussed the assignment with.
  – No written material may be exchanged during the discussion. If you talk with someone about the assignment, you must throw away and written notes at the end of the discussion.
Problem 1 (20 points): We wish to implement a dictionary by using direct addressing on a huge array. At the start, the array entries may contain garbage, and initializing the entire array is impractical because of its size. Describe a scheme for implementing a direct-address dictionary on a huge array. Each stored object should use $O(1)$ space; the operations SEARCH, INSERT, and DELETE should take $O(1)$ time each; and initializing the data structure should take $O(1)$ time. (Hint: Use an additional array, treated somewhat like a stack whose size is the number of keys actually stored in the dictionary, to help determine whether a given entry in the huge array is valid or not.)
Problem 2 (20 points): Demonstrate what happens when we insert the keys 5, 28, 19, 15, 20, 33, 12, 17, 10 into a hash table with collisions resolved by chaining. Let the table have 9 slots, and let the hash function be $h = k \mod 9$.

Show the solution by showing which slot each key goes to, and what order the items are in the linked list.
Problem 3 (20 points): Consider inserting the keys 10, 22, 31, 4, 15, 28, 17, 88, 59 into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k$. Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (kmod < m - 1)$.