CSC 228 Data Structures and Algorithms, Fall 2017
Instructor: Dr. Natarajan Meghanathan

Project 5: Using Hashtables to Print the Duplicate Elements in an Array exactly once

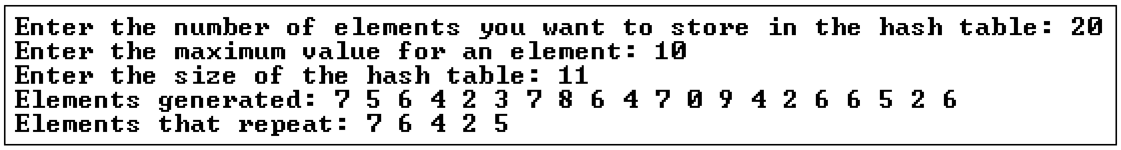
Due: October 18, 1 PM

The objective of this project is to design and implement a Θ(n) algorithm (using hash tables) to print exactly once the duplicate elements (the elements that appear more than once) in an array of integers.

Use the code given for the hash table class (implemented using singly linked list) and the startup code for the main function to generate an array of integers. Test your code with the following values:
array size: 20, maximum value for an integer: 10 and hash table size: 11.

Implement the algorithm in the main function itself in the space provided.

Note that your algorithm should print the duplicate elements only once. For example, the following is the output for an array of random integers generated and hashed with the above parameter values. Note that even though the (duplicate) integers 7, 6, 4, 2, 5 appear more than once, they are printed only once as part of the output for the elements that repeat.



```
Enter the number of elements you want to store in the hash table: 20
Enter the maximum value for an element: 10
Enter the size of the hash table: 11
Elements generated: 7 5 6 4 2 3 7 8 6 4 7 0 9 4 2 6 6 5 2 6
Elements that repeat: 7 6 4 2 5
```

Space complexity: You are free to use more than one hash table of the size input by the user for this problem.

Submission (through Canvas):
(1) Complete code of the main function implementing the algorithm as well as the code for the Hashtable, Node and Singly Linked List classes.
(2) A report: explain the logic behind your algorithm (that it indeed finds duplicate elements and also prints them exactly once), provide a pseudo code of the algorithm and justify that the time complexity of the algorithm is Θ(n). Assume the time complexity to search for the presence/absence of an element in a Hashtable is Θ(1).
(3) Explain the purpose of each hash table used to accomplish the Θ(n) time complexity.
(4) A screenshot of the output (similar to the sample provided above) of running your code for the main function after implementing the algorithm.