**CS311 DSA Homework 4**

**Your Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Univ. ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Instructions**

* Check **due date** and polices on course webpage.
* Write the answer for each question in the space provided below the question.
* Submit your answers to cs\_scu@foxmail.com.
* Submission file name format: CS311\_assignmentID\_yourID\_yourName.doc (or .pdf).

1. (10 points) Write 1) a *sequential search* algorithm to find any positions in an unsorted array *A* that holds value *K*, and 2) a *binary search* in a sorted array *A*. (Assume the length of the array *A* is *n*.)

***Answer***:

2. (15 points) For each of the following three self-organization list heuristics, describe a series of record accesses for which it would require the greatest number of comparisons of the three.

1. Move-to-Front
2. Count
3. Transpose

***Answer***:

3. (20 points) Assume that you are hashing key *K* to a hash table of *n* slots (index from 0 to *n*-1). For each of the following functions *h*(*K*), is the function acceptable as a hash function (i.e., would the hash program work correctly for both insertions and searches), and if so, is it a good hash function, and why? (Note, consider the problem of *collision*.)

1. *h*(*k*)=1
2. *h*(*k*)=*k/n*, where *k* and *n* are integers
3. *h*(*k*)=(*k+Random*(*n*)) mod *n*, where Function *Random*(*n*) returns a random integer between 0 and *n*-1
4. *h*(*k*)=*k* mod *n*, where *n* is a prime number

***Answer***:

4. (25 points) Here are five potential implementations of the Integer class's hashCode() method. Categorize each as (1) invalid, (2) valid but not good, and (3) valid and good. If it is invalid, explain why. If it is valid but not good, point out a flaw or disadvantage. For the 2nd implementation, node that intValue() will return that Integer's number value as an int, and assume that Integer's equals method checks for equality of the compared Integers' intValues.

a.

**public int** hashCode() {

 **return** -1;

}

b.

**public int** hashCode() {

 **return** intValue()\*intValue();

}

c.

**public int** hashCode() {

 **return super**.hashCode(); // Object's hashCode() is based on memory location

}

d.

**public int** hashCode() {

 **return (int)** (**new** Date()).getTime(); // returns the current time as an int

}

e.

**public int** hashCode() {

 **return** inValue() + 3;

}

***Answer***:

4. (20 points) Write the graph below as an adjacency matric, then as an adjacency list. What would be different if the graph were undirected instead?

***Answer*:**

5. (10 points) Given an undirected graph, provide an algorithm that returns true if a cycle exists in the graph, and false otherwise. Also, provide a Θ bound for the worse case runtime of your algorithm. (your idea, pseudocode, and bound result)

***Answer*:**