Data Structure and algorithm
Lab 01: Array

1. **Objectives**
   a. Review of Array class, object, and operations (construct, find, delete, insert).
   b. Know how, in reality, Array works.

2. **Problem statement**
   a. Run the Array Workshop Applets (Array & OrderedArray) to deal with Array operations
   b. Compile and Run the Example Programs
      i. Array, LowArray, HighArray
      ii. OrderedArray
      iii. ClassDataArray
   c. Programming Projects 2.1 in Text-Book

      To the *HighArray class* in the *highArray.java* program *(Listing 2.3)*, add a method called *getMax()* that returns the value of the highest key in the array, or –1 if the array is empty. Add some code in *main()* to exercise this method. You can assume all the keys are positive numbers.
   d. Programming Projects 2.2 in Text-Book

      Modify the method in Programming Project 2.1 so that the item with the highest key is not only returned by the method, but also removed from the array. Call the method *removeMax()*.
   e. Programming Projects 2.5 in Text-Book

      Add a *merge()* method to the *OrderedArray class* in the *orderedArray.java* program *(Listing 2.4)* so that you can merge two ordered source arrays into an ordered destination array. Write code in *main()* that inserts some random numbers into the two source arrays, invokes *merge()*, and displays the contents of the resulting destination array. The source arrays may hold different numbers of data items. In your algorithm you will need to compare the keys of the source arrays, picking the smallest one to copy to the destination. You'll also need to handle the situation when one source array exhausts its contents before the other.
   f. Programming Projects 2.6 in Text-Book

      Write a *noDups()* method for the *HighArray class* of the *highArray.java* program *(Listing 2.3)*. This method should remove all duplicates from the array. That is, if three items with the key 17 appear in the array, *noDups()* should remove two of them. Don't worry about maintaining the order of the items. One approach is to first compare every item with all the other items and overwrite any duplicates with a null (or a distinctive value that isn't used for real keys). Then remove all the nulls. Of course, the array size will be reduced.

3. **Simple application**
   a. Write a function to convert array to number.

      Suppose we have loaded an array with the digits of an integer,

      where the highest power is kept in position 0, next highest in position 1, and so on.

      The ones position is always at position array.Length - 1:

      int[] digits = { 2, 0, 1, 8 };
   b. Write a function to input a list of integer numbers and return the median of that list.

Revised by: Tran Thanh Tung