CS272 Lab Assignment #13: Heap, Search, and Hash

Learning objectives: Objective 1 (heap, hash), Objective 2 (recursive thinking), Objective 3 (searching), Objective 5, Objective 6, Objective 7

Note:

- **Specifications** for all your classes and methods:
  Please properly explain (1) the functionality of the methods, (2) the parameters, (3) the return values, (4) the pre-conditions if there is any;
  Please use inline comments, meaningful variable names, indentation, formatting, and whitespace throughout your program to improve its readability.
- You can (but are not required to) design and implement other facilitating methods (E.g., other get and set methods, toString method) to finish the implementation of the required methods.

Requirements

1. (40 pts) Implement the following methods for a min heap (i.e., the top element of the heap is the smallest element) by using the *ArrayList* java class to store the heap elements.
   Put all your methods to **MinHeap.java**.
   - (15 pts) Add a new element *e* into the heap.

     ```
     public void add(int e)
     ```

   - (15 pts) Get and remove the top element of the heap.

     ```
     public int remove()
     ```

   - (10 pts) Get the top element of the heap.

     ```
     public int top()
     ```

   - Put your test cases to the main method. You need to design test cases to test your program *thoroughly*. If your test cases cannot cover some important conditions, points may be deducted.

2. (20 pts) Please design the binary search function to search an element `e` in an array `A`. Assume that all the elements in array `A` are useful elements, and the values in `A` are ordered in ascending order.
   Put your test cases to the main method. You need to design test cases to test your program *thoroughly*. If your test cases cannot cover some important conditions, points may be deducted.
   Put all your method and test code to **search.java**.

   ```
   public static int binarySearch (int[] A, int e)
   ```

3. (40 pts) Please design a hash table `BookTable` to implement the

open-address hashing data structure that we discussed in class. `BookTable` is used to store information of books in a store. The book structure is built upon the data structure you implemented in lab 1. For this lab, each book should also have a String variable `ISBN`, which is a unique identifying value of a book.
Implement the following methods for this class and put the code to **BookTable.java**.

- (2 pts) Please include proper instance variables in `BookTable`.
- (2 pts) Please include proper constructors.
- (2 pts) Please design the hash function to be

  ```
  the hash code of the ISBN % size_of_array_for_keys
  ```

- (12 pts) Add a new book *e* into the hash table.

  ```
  public void put(Book e)
  ```

- (12 pts) Remove a given book with id *ISBN* from the hash table. Return false if a book with this isbn does not exist in the hash table; Otherwise, remove it and return true.

  ```
  public int remove(int isbn)
  ```

- (10 pts) Find the index of the given book isbn. Return the index of the book in the array if the book with the given isbn exists in the hash table; otherwise, return -1.

  ```
  public int search(int isbn)
  ```

- Put your test cases to the main method. You need to design test cases to test your program *thoroughly*. If your test cases cannot cover some important conditions, points may be deducted.

Submission:

a zipped file *your-bannerid-lab13.zip* containing your java file(s).

Grading Criteria

- The score allocation has already been put beside the questions.
- Please make sure that you test your code thoroughly by considering all possible test cases.
  Your code may be testd using more test cases.