Lab Tasks 3

1. Add the following method to the class hashing.QuadraticProbingHashing.

   ```
   public AnyType[] elements();
   ```

   This method should return an array that contains all items currently present in the hashtable, and no **null** values.

2. Modify the class heap.BinaryHeapUnique such that only at most one copy of *equal* elements are included in the heap. Recall that two items item1 and item2 are considered *equal* iff the call

   ```
   item1.compareTo(item2)
   ```

   returns 0.

   Thus the call of compareTo in the following program will always return **false**.

   ```
   BinaryHeapUnique<SomeType> bh = new BinaryHeapUnique<SomeType>;
   ....
   SomeType item1 = bh.deleteMin();
   SomeType item2 = bh.deleteMin();
   System.out.println(item1.compareTo(item2));
   ```

3. Recall that the length of a path in a tree is defined as its number of edges, and that the depth of a node n is the length of the path from the root to n. Complete the method

   ```
   public int depth(AnyType x)
   ```

   in BinaryHeapWithDepth such that it returns the smallest depth of a node that contains an item equal to x.

4. Modify the class search.BinarySearchTree such that it keeps track of multiple copies of elements. The method insert adds a copy, and the method remove removes a copy (if it is present). Two elements item1 and item2 are considered copies of the same element, iff

   ```
   item1.compareTo(item2)
   ```

   returns 0. The method contains should return **true** if there is at least one copy of the element.

   Complete the method instances which should return the number of copies of a given element that is currently contained in the tree.