**Programming and Data Structures**
**Active Learning Activity 7: Recursion and Generics**

**Activity Objectives**

At the end of this activity, students should be able to:

1. Write a recursive method to solve a given recursive problem
2. Create a generic class with two generic types
3. Use Java generic class **`ArrayList`**
4. Create a generic method to search in a generic ArrayList

**Activity**

---

1. Write a recursive method **`int getSize(String directoryName)`** that returns the total size of all the files under the directory **`directoryName`**. Use the class **`File`** methods **`isDirectory()`** and **`isFile()`** to find if a pathname is a directory or a file. Use the method **`length()`** to get the size of a file in bytes. Display the total size with the appropriate suffix as shown in the examples below.
   If total size = 52,000 bytes, the program should display "52 Kbytes"
   If total size = 2,000,000 bytes, the program should display "2 Mbytes"
   If total size = 9,000,000,000 bytes, the program should display "9 Gigabytes"

2. Write a test program (**`Test.java`**) to test your recursive method. In the main method, prompt the user to enter a pathname and display the total size of the directory. If the pathname is a file, the program should display the size of the file. If the pathname is neither a directory nor a file, the program should display 0.

3. Create the generic class **`Pair<E1, E2>`** with two generic types as seen in class. The class should have appropriate constructors, accessors, mutators, **`toString()`**, and **`equals()`** methods as seen in class.

4. In the same main method from step 2, create an instance of the generic class **`ArrayList`** for **`Pair<String, Integer>`** and name it **`listTrees`**. Each

element in **listTrees** contains the name of a tree and its maximum height in feet. Add the following information to **listTrees** for five different trees.

```
"Leather Leaf Acacia", 12 ft
"Key Lime", 24 ft
"American Hazelnut", 24 ft
"Flowering Maple", 24 ft
"Silverberry", 36 ft
```

5. Create another instance of the generic class **ArrayList** for **Pair<Integer, Double>** and name it **listStudents**. Each element in **listStudents** contains student ID and GPA of a student. Add the following information to **listStudents** for five different students.

```
(12345, 3.96)
(54321, 2.25)
(12453, 3.50)
(53421, 2.83)
(51234, 1.25)
```

6. Define a generic method **search** in class **Test** with the header below.

```
public static <E1, E2> int search(
                        ArrayList<Pair<E1, E2>> list, E1 key)
```

**search** returns the index where **key** is found in **list** or -1 otherwise. Note that **key** is compared to the first element of the pair.

7. Prompt the user to select the type of search she/he would like to perform (1: search for a tree or 2: search for a student). Prompt the user to enter a tree name or a student ID and call **search()** to look in the appropriate list. Display the full information of the element that matches user input or the message "`Item not found`". Re-prompt the user for another search until the user decides to stop.

8. Do not forget to include `Javadoc` comments in your code. Submit your files **Pair.java** and **Test.java** on coursesite.