**44-542 Object Oriented Programming Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Exam 03 Part 2 (40 points) KEY** *please print*

1. (5 pts) Suppose we are using the hash function **h(key) = key MOD 10** to store records with keys **11, 21, 31, 89, 99, 59**. Show where each key is stored in the table below. Use chaining to resolve collisions. You must apply **h** to the keys in the order in which they are listed.

|  |  |
| --- | --- |
| **Index** |  |
| **0** |  |
| **1** | **list 🡪 11 🡪21 🡪 31 🡪 null** |
| **2** |  |
| **3** |  |
| **4** |  |
| **5** |  |
| **6** |  |
| **7** |  |
| **8** |  |
| **9** | **list 🡪 89 🡪 99 🡪 59 🡪 null** |

1. (7 pts) Write a code segment to do the following: Create a **HashSet** of **String** objects named **myFriends** and add “Sue” and “David” to the hash set. No class headings or import statements are needed.

**HashSet<String> myFriends = new HashSet<String>();**

**myFriends.add("Sue");**

**myFriends.add("David");**

1. (5 pts) For the tree below, show the order in which the nodes are visited in a postorder traversal.



Answer: \_\_\_\_\_\_**7 15 25 40 50 30 20**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (10 pts) Suppose we have a class **Books**, as shown below. The tree map **bookList** contains publisher names as key values (Avon Press, Berkley Press, etc.). The value associated with a key is the list of books published by that publisher. Method **addBook** has two arguments – a string representing the name of a publisher, and a string representing the title of a book. This method will add this key-value pair to the tree map.

For example, we might have the following code in a driver program:

**Books myBooks = new Books();**

**myBooks.addBook("Avon Press", "Java 5");**

**myBooks.addBook("Berkley Press", "C#");**

**myBooks.addBook("Avon Press", "Python");**

After this code is executed, we would have two entries in the tree map:

**Avon Press = [Java 5, Python]** and **Berkley Press = [C#]**

Write the missing code for **addBook** below.

**public class Books**

**{**

**private TreeMap<String, ArrayList<String>> bookList;**

**public Books()**

**{**

**bookList = new TreeMap<String, ArrayList<String>>();**

**}**

**public void addBook(String publisher, String bookTitle)**

**{**

**ArrayList<String> bookTitles = bookList.get(publisher);**

**if(bookTitles == null)**

**{**

**bookList.put(publisher, new ArrayList<String>());**

**}**

**bookList.get(publisher).add(bookTitle);**

**}**

**}**

1. (8 pts) Consider the following tree:



* 1. What is the height of this tree? \_\_\_\_\_\_\_3 \_\_\_\_\_
  2. How many leaves does this tree have? \_\_\_\_\_\_\_5 \_\_\_\_\_\_\_\_\_\_\_
  3. What is the left child of 20? \_\_\_\_\_\_\_15 \_\_\_\_\_\_\_\_\_\_\_
  4. How many probes are necessary to discover that 17 is not in the tree? \_\_4\_\_\_\_\_

1. (5 pts) Using the algorithm given in class, insert the following nodes into a binary search tree. The nodes must be inserted in the order given here.

**45 50 18 20 60**



1. (5 pts) Using the algorithm given in class, remove 160 from the binary search tree given below.





1. (5 pts) Using the algorithm given in class, perform a left rotation around 80.





1. (5 pts) Using the algorithm given in class, insert the following nodes into a heap. The nodes must be inserted in the order given here.

**90 85 80 75 70 65 60**



1. (5 pts) Using the algorithm given in class, remove the top element from the heap shown below.



