**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 linear probing to resolve collisions. You must apply **h** to the keys in the order in which they are listed.

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

1. (7 pts) Write a code segment to do the following: Create a **TreeSet** of **String** objects named **mySiblings** and add “Bob” and “Amy” to the tree set. No class headings or import statements are needed.

**TreeSet<String> mySiblings = new TreeSet<String>();**

**mySiblings.add("Bob");**

**mySiblings.add("Amy");**

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



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

1. (10 pts) Suppose we have a class **Enrollment**, as shown below. The tree map **enrollment** contains course numbers as key values (44542, 44560, etc.). The value associated with a key is the list of students enrolled in the course. Method **addEnrollment** has two arguments – a string representing a course number, and a string representing the name of a student. This method will add this key-value pair to the tree map.

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

**Enrollment myEnrollments = new Enrollment();**

**myEnrollments.addEnrollment("44542", "Amy");**

**myEnrollments.addEnrollment("44560", "Bill");**

**myEnrollments.addEnrollment("44542", "Sue");**

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

**44542 = [Amy, Sue]** and **44560 = [Bill]**

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

**public class Enrollment**

**{**

**private TreeMap<String, LinkedList<String>> enrollment;**

**public Enrollment()**

**{**

**enrollment = new TreeMap<String, LinkedList<String>>();**

**}**

**public void addEnrollment(String course, String student)**

**{**

**LinkedList<String> students = enrollment.get(course);**

**if(students == null)**

**{**

**enrollment.put(course, new LinkedList<String>());**

**}**

**enrollment.get(course).add(student);**

**}**

**}**

1. (8 pts) Consider the following tree:



* 1. How many leaves does this tree have? \_\_\_\_\_\_\_4 \_\_\_\_\_
  2. What is the height of this tree? \_\_\_\_\_\_\_3 \_\_\_\_\_\_\_\_\_\_\_
  3. What is the right child of 30? \_\_\_\_\_\_\_50 \_\_\_\_\_\_\_\_\_\_\_
  4. How many probes are necessary to discover that 45 is not in the tree? \_\_5\_\_\_\_\_

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.

**15 10 12 22 18**



1. (5 pts) Using the algorithm given in class, remove 32 from the binary search tree given below.





1. (5 pts) Using the algorithm given in class, perform a right 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.



**25 20 15 10 5 2 1**

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



