Lee Moriarity

Professor Kelly

CIS 112 ONL01

6 November 2022

Lab #8

Do the following exercises  at the end of Chapter 8 (starting at p. 544):

**Exercise #6, #14, #15, #17, #20, #21**

Exercise #6:

6. What is the output of the following code—ArrayListMap is an implementation of

the MapInterface and is presented in Section 8.2:

MapInterface<Character, String> question;

question = new ArrayListMap<Character, String>();

System.out.println(question.isEmpty());

System.out.println(question.size());

System.out.println(question.put('M', "map"));

question.put('D', "dog"); question.put('T', "Top");

question.put('A', "ant"); question.put('t', "Top");

System.out.println(question.isEmpty());

System.out.println(question.size());

System.out.println(question.contains('D'));

System.out.println(question.contains('E'));

System.out.println(question.get('D'));

System.out.println(question.get('E'));

System.out.println(question.put('D', "dig"));

System.out.println(question.get('D'));

for (MapEntry<Character,String> m: question)

System.out.print(m.getValue() + "\t");

System.out.println(question.remove('D'));

System.out.println(question.remove('D'));

for (MapEntry<Character,String> m: question)

System.out.print(m.getValue() + "\t");

Output:

true

0

null

false

5

true

false

dog

null

dog

dig

map Top ant Top dig dig

null

map Top ant Top

Exercise #14:

14. For each of the following array sizes indicate whether or not the following sequence

of key insertions 21, 75, 240, 413, 1368, 9021, 9513 using a hashing system causes a

collision when the appropriate compression function is used:

a. 10 collision

b. 15 collision

c. 17 no collision

d. 21 collision

e. 50 collision

f. 100 collision

g. 1,000 collision

h. 10,000 no collision

Exercise #15:

15. Show the array that results from the following sequence of key insertions using a

hashing system under the given conditions: 5, 205, 406, 5205, 8205, 307 (you can

simply list the non-null array slots and their contents)

a. array size is 100, linear probing used.

b. array size is 100, quadratic probing used.

c. array size is 101, linear probing used.

d. array size is 1,000, linear probing used.

e. array size is 100, with buckets consisting of linked lists.

Exercise #17:

17. Show the array that results from the following sequence of operations using a hashing

system under the given conditions: insert 5, insert 205, insert 406, remove 205,

insert 407, insert 806, insert 305 (you can simply list the non-null array slots and

their contents)

a. array size is 100, linear probing used, the value 1 indicates a removed entry.

b. array size is 10, linear probing used, the value 1 indicates a removed entry.

c. array size is 10, linear probing used, an additional value of type boolean is

used, with true indicating a slot has been used.

d. array size is 10, with buckets consisting of linked lists.

Exercise #20:

20. In the nine-digit city code example suppose the hash function used is [(A x C) + B].

What is the hash code for each of the following cities?

a. Hangzhou, Zhejiang, China: 001112038

**[(11 x 38) + 12] = 430**

b. Lancaster, Pennsylvania, USA: 012113103

**[(121 x 103) + 13] = 12476**

c. Yiyang, Hunan, China: 321732038

**[(3217 x 38) + 32] = 122278**

d. Beaver Falls, Pennsylvania, USA: 54213103

**[(5421 x 103) + 31] = 558394**

e. Seoul, Seoul, South Korea: 010313121

**[(103 x 121) + 13] = 12476**

Exercise #21:

21. Critique the following hash functions for a domain consisting of people with attributes

  firstName, lastName, number (used to resolve identical first and last

names, e.g., “John Smith 0,” “John Smith 1,” etc.), and age. The names are of class

String and the other two attributes are of type int.

a. hash function returns (age)2

**public int hashCode() { return age \* age;}**

b. hash function returns (age)2 + lastName.hashCode()

**public int hashCode() { return (age \* age) + lastName.hashCode();}**

c. hash function returns lastName.hashCode() + firstName.hashCode()

**public int hashCode() { return lastName.hashCode() + firstName.hashCode();}**

d. hash function returns lastName.hashCode() + firstName.hashCode() + number

**public int hashCode() { return lastName.hashCode() + firstName.hashCode() + number;}**