15-121: Midterm Exam

Summer 2011

SOLUTIONS

1. (3 pts.) In the context of hashing, what is meant by a collision?
   a) The hash function returns a value index greater than the table size.
   b) The insertion algorithm cannot find an empty slot in the table.
   c) Two equal objects hash to the same slot.
   d) Two unequal objects hash to the same slot.

2. (3 pts.) Which of the following features is not related to object-oriented programming?
   a) Garbage collection
   b) Encapsulation
   c) Polymorphism
   d) Inheritance

3. (3 pts.) You work for a very busy retail store. You are asked to construct a list of customer names: a customer's name is added to a list each time s/he enters the store, if it isn't already there. What data structure would be best-suited to hold the list?
   a) Singly Linked List
   b) Hash Table
   c) Stack
   d) FIFO Queue

4. (3 pts.) Which of the following data structures allows us to find any element in $O(1)$ expected runtime?
   a) Stack
   b) Queue
   c) Hash Table
   d) Linked List

5. (3 pts.) Which of the following are true?
   a) If x.equals(y) is true, then x.hashCode() == y.hashCode()
   b) If x.hashCode() == y.hashCode(), then x.equals(y) is true
   c) Both A and B are true
   d) Neither A nor B are true

6. (5 pts.) When would you use the equals() method instead of ==?
   a) It doesn't matter, they are interchangeable.
   b) You should always use equals().
   c) equals() should be used to compare primitive data types.
   d) equals() should be used to compare Objects

7. (10 pts.) Predict the output for each of the following statements

   a) "Yesterday".substring(5,5).length() _____0_____

   b) "Tomorrow".indexOf("o", 2) _____3_____

   c) "Yahoo".split("o")[0]; _____"Yah" _____

   d) "Google".split("o")[1]; _____ empty string ____

   e) "Google".split("o+")[1]; _____"gle" _____

8. (5 pts.) Given an efficient circular array-based queue q capable of holding 7 objects. Show the final contents of the array after the following code is executed:

```
for (int k = 1; k <= 6; k++)
   q.enqueue(k);

for (int k = 1; k <= 3; k++)
{
   q.dequeue();
   q.enqueue(q.dequeue());
}
```

0	1	2	3	4	5	6
4	6					2

9. (5 pts.) Given a circular array-based queue capable of holding 100 objects. Suppose the queue is initially empty, and then objects are put into the queue at the rate of 10 per minute while meantime they are processed and removed from the queue at the rate of 5 per minute. After 120 elements have been added to the queue, which of the following is true?

a) You can't add 120 elements to an array holding 100 entries.

b) There will be 60 elements in the queue, 20 of them at the front of the array where the queue started, and 40 at the other end.

c) There will be 60 elements in the queue, 30 of them at the front of the array where the queue started, and 30 at the other end.

d) There will be 60 elements in the queue, 40 of them at the front of the array where the queue started, and 20 at the other end.

10. (10 pts) Insert the following sequence of numbers

23, 46, 12, 21, 75, 5, 3

into a hash table of size 9 using h(x) = x%9 as a hash function. Use a linear probing for resolving collisions.

0	1	2	3	4	5	6	7	8
	46		**12**	**21**	**23**	**75**	**5**	**3**

11. (10 pts.) Examine the following code segment

```
Set<String> st = new HashSet<String>();
for (int k = 0; k < 4; k++) st.add("ab" + (k%2) );
Iterator<String> itr = st.iterator();
int n = 0;
while(itr.hasNext())
{
    n += itr.next().length();
}
System.out.println(n);
```

What does this code print?

a) 12

b) 6

c) 3

d) ab12

12. (12 pts.) What will the following code print?

```
public class ChangePlaces
{
   private int numOne = 5;
   public static int numTwo = 0;

  public ChangePlaces()  {
     int temp = numTwo;
     numTwo = numOne;
     numOne = temp;
  }

  public void increment()   {
     numOne++;
     numTwo++;
  }

  public int getNumOne()  {return numOne;}}

public class RunMe   {
    public static void main(String[] args)     {
        ChangePlaces foo = new ChangePlaces();
        foo.increment();
        ChangePlaces bar = new ChangePlaces();
        bar.numTwo = 100;
        ChangePlaces baz = new ChangePlaces();
```

```
        System.out.println("foo: numOne = " + foo.getNumOne() + ", numTwo = " + foo.numTwo);
        System.out.println("bar: numOne = " + bar.getNumOne() + ", numTwo = " + bar.numTwo);
        System.out.println("baz: numOne = " + baz.getNumOne() + ", numTwo = " + baz.numTwo);

    }
}
```

1) foo: numOne = ____**1**_____, numTwo = _____**5**_____
2) bar: numOne = _____**6**_____, numTwo = _____**5**_____
3) baz: numOne = _____**100**_____, numTwo = _____**5**_____

13. (8 pts) Implement a method that returns the total sum of all elements of a given ArrayList of integers using an **Iterator**

```
public int sumUp( ArrayList<Integer>  list )
{
   int sum = 0;
   Iterator<Integer> itr = list.iterator();
   while( itr.hasNext() )
   {
      sum += itr.next() );
   }
  return sum;
}
```

14. (10 pts.) Write a method that takes a Map and prints the keys in a sorted order.

```
public void printKeys(Map<Integer, List<Integer>> data)
   {
      ArrayList<Integer> list = new ArrayList<Integer>( map.keySet());
      Collections.sort(list);
      System.out.println(list);
```

15. (10 pts.) Fill in the blank in the following implementation of the remove() method for a non-circular doubly linked list.

```
import java.util.NoSuchElementException;
public class DoublyLinkedList<AnyType>
{
    private Node<AnyType> head;

    public DoublyLinkedList() {
        head = null;
    }

    /*
     * Removes the Node with the specified data from the list.
     * If there is more than one Node with that data, remove the first one.
     * If there are no Nodes with that data, throw a NoSuchElementException.
     */
    public void remove(AnyType key)
    {
        if(_____head == null_____) throw new NoSuchElementException("Element does not exist.");
        else if(head.data.equals(key))
        {
            head = ____head.next; _____________;
            return;
        }

        Node<AnyType> cur = head;
        while(___ cur != null && !cur.data.equals(key)  _____________) cur = cur.next;
        if(_ cur == null ____) throw new NoSuchElementException("Element does not exist.");

        cur.prev.next = cur.next;
        cur.next.prev = cur.prev;     }

    private static class Node<AnyType>
    {
        private AnyType data;
        private Node<AnyType> next, prev;

        public Node(AnyType data, Node<AnyType> next, Node<AnyType> prev) {
            this.data = data;
            this.next = next;
            this.prev = prev;
        }
    }
}
```