Computing & Algorithms II

CS-102 **Fall 2008**

Midterm Exam **Solution Set**

1. *5 points each. Give* brief *definitions of the following concepts:*

   (a) *Abstract data type.* A collection of data elements and operations that can be performed on those elements.

   (b) *Queue.* A list accessible in a first-in, first-out manner. Items are appended to one end of the list and removed from the other end.

   (c) *Exception.* An object representing an unusual or erroneous run-time condition.

2. *10 points each. For each of the following pairs of items, describe an advantage each has over the other. (*E.g., *"Pepsi tastes better, but Coke is cheaper.")*

   (a) *Dynamic linked lists versus array-based lists.* Dynamic linked lists are not limited by size, use only the space required, and require less bulk copying ("shifting") of elements for add and remove operations; array-based lists permit random access, take less space for stable lists (due to less overhead), and do not require dynamic memory.

   (b) *Doubly-linked lists versus "ordinary" linked lists.* Doubly-linked lists permit bi-directional movement and make add/remove operations more straightforward (due to immediate access to predecessors); ordinary linked lists require less memory and fewer reference updates to perform add and remove operations.

3. *10 points each. Consider the following definitions (similar to those given in lecture):*

```
public class ListNode {
     private int datum;
     private ListNode next;

     public ListNode() { datum = 0; next = null; }
     public ListNode(int val) { datum = val; next = null; }

     public int getDatum() { return datum; }
     public ListNode getNext() { return next; }

     public void setDatum (int val) { datum = val; }
     public void setNext (ListNode val) { next = val; }
}
```

```
public class LinkedList {
     private ListNode head;
}
```

Consider the following Java methods belonging to the ***LinkedList*** *class. All of the methods compile correctly; however, they do not function as described in their comments. What is wrong? How would you fix them?*

(a)
```
// These methods recursively print all of the integers in the list

public void printList () {
    printList(head);
}
private void printList (ListNode current) {
    if (current != null) {
        System.out.println(current);
        printList(current.getNext());
    }
}
```

This prints each node, not the contents of each node. To fix this, replace "`current`" by "`current.getDatum()`".

(b)
```
// This method searches for a specified object in the list,
// returning true if found, false if not.

public boolean search (int target) {
    return search (target, head);
}

private boolean search (int target, ListNode current) {
    if (current.getDatum() == target)
        return true;
    return search (target, current.getNext());
}
```

This method never returns "`false`", because one of the base cases is ignored. To fix this, add the following code to the beginning of the private method:

```
             if (current==null) return false;
```

(c)
```
// This method adds the object at the
// specified position (0 = head, 1 = after head, etc.)
```

```
public void add (int value, int index) {
    ListNode current = head;            // current pos. in list
    ListNode prev = null;               // trailing pointer

    while ((index > 0) && (current != null)) {
        prev = current;                 // take a step
        current = current.getNext();
        index--;
    }
    if (current == null)                   // ran out of list
        throw new IndexOutOfBoundsException();

    ListNode splice = new ListNode(value);  // new node
    if (prev == null)
        head = splice;
    else prev.setNext(splice);             // set up links
    splice.setNext(current);               // for new node
}
```

The test after the loop is incorrect; replace “`(current==null)`” with “`(index != 0)`”.

4. *20 points. Write a Java method in the **`LinkedList`** class with the following header:*

   `public int count (int target)`

   *This method should return the number of occurrences of “**`target`**” in the list.*

```
public int count (int target) {
    ListNode current = head;
    int answer = 0;
    while (current != null) {
        if (current.getDatum() == target)
            answer++;
        current = current.getNext();
    }
    return answer;
}
```

5. *15 points. The* triangular numbers T_n *satisfy the following formula:*

$$T_n = 1 + 2 + 3 + \ldots + (n-1) + n$$

That is, for any n, T_n is the sum of the first n positive integers.

Write a recursive *Java method which, given input n, computes and returns the nth triangular number T_n.*

```
public int triangular (int input) {
    if (input == 1)
        return 1;
    return input + triangular(input - 1);
}
```