Com S 228

Spring 2018

Exam 2 Sample Solution

1. Element returned by the last next() or previous() is underlined. The final state is in bold.

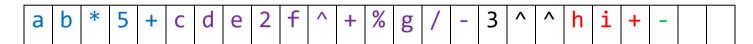
Code snippet	Output	List and iterator state, or exception thrown
<pre>iter = aList.listIterator();</pre>	(none)	A B X C D
<pre>iter = aList.listIterator(2); System.out.println(iter.previous());</pre>	В	АВ ХС D А <u>В</u> ХС D
<pre>iter = aList.listIterator(aList.size()); iter.remove();</pre>	(none)	IllegalStateException
<pre>iter = aList.listIterator(); while (iter.hasNext()) { iter.set(iter.next() + iter.previous()); System.out.println(iter.next()); }</pre>	AA BB XX CC DD	ABXCD AA BB XX CC DD
<pre>iter = aList.listIterator(); while (iter.hasNext()) { iter.add(iter.next()); System.out.println(iter.previous()); iter.next(); }</pre>	A B X C	iteration 1: A B X C D A A B X C D A A B X C D A A B X C D iterations 2 to 5: A A B B X X C C D D
<pre>iter = aList.listIterator(); iter2 = aList.listIterator(aList.size()); while (iter.nextIndex() < iter2.previousIndex()) { String s = iter.next(); String t = iter2.previous(); iter.set(t); iter2.set(s); }</pre>	(none)	A B X C D iteration 1: A B X C D A B X C D D B X C A iteration 2: D C X B A
<pre>iter = aList.listIterator(); iter2 = aList.listIterator(1); while (iter2.hasNext()) { iter.next();</pre>	B X C D	A B X C D iteration 1:

2. a) O(n); b) O(1); c) O(n+k).

3a) Infix:

$$(a * b + 5) ^ (c - d % (e + 2 ^ f) / g) ^ 3 - (h + i)$$

Postfix:



b) Postfix:

```
1 a 2 b 3 c 4 d 5 e / + / + / + / + /
```

Infix:

```
//
// Let list3.head reference the smallest item from the two lists.
// Note that one or both lists may be empty.
if (cur1 != null && cur2 != null)
      if (comp.compare(cur1.data, cur2.data) <= 0)</pre>
      {
            list3.head = new Node (cur1.data, null);
            cur1 = cur1.next;
      }
      else
      {
            list3.head = new Node (cur2.data, null);
            cur2 = cur2.next;
      }
else if (cur1 != null)
      list3.head = new Node (cur1.data, null);
      cur1 = cur1.next;
else if (cur2 != null)
      list3.head = new Node (cur2.data, null);
      cur2 = cur2.next;
}
// 2. Merging (10 pts)
// Iterate the two references cur1 and cur2 through the remaining nodes.
// Construct list3 on the fly by creating new nodes using the reference
// cur3.
Node cur3 = list3.head; // This variable is used for generating nodes of
                        // list3.
// Both lists have unprocessed elements.
while (cur1 != null && cur2 != null)
      if (comp.compare(cur1.data, cur2.data) <= 0)</pre>
      {
            cur3.next = new Node (cur1.data, null);
            cur1 = cur1.next;
      }
      else
      {
            cur3.next = new Node (cur2.data, null);
            cur2 = cur2.next;
      cur3 = cur3.next;
}
```

```
// 3. List appending (10 pts)
      //
      // Step 2 stops when the end of one list is reached (thus all of its items
      // have been added to list3 at this point). Append the remainder of the
      // other list to list3.
      // All the items from list1 have been added to list3.
      if (cur1 == null)
      {
            while (cur2 != null)
                  cur3.next = new Node(cur2.data, null);
                  cur2 = cur2.next;
                  cur3 = cur3.next;
            }
      }
      // All the items from list2 have been added to list3.
      else // cur2 == null
      {
            while (cur1 != null)
            {
                  cur3.next = new Node(cur1.data, null);
                  cur1 = cur1.next;
                  cur3 = cur3.next;
            }
      }
      // 4. Updating instance variables, if any, of the merged list. (4 pts)
      list3.size = list1.size + list2.size;
      return list3;
}
```