Com S 228

Spring 2014

Exam 2 Sample Solution

1a) O(1); b) $O(n)$; c) O(1); d) $O(n)$; e) $O(n^2)$

2.

Code snippet	**Output**
`iter = aList.listIterator();` `print(aList, iter.nextIndex());`	`| A B C D E`

`// 3 pts`
`iter = aList.listIterator(3);`
`System.out.println(iter.next());`
`print(aList, iter.nextIndex());`
	`D`
	`A B C D | E`

`// 5 pts`
`iter = aList.listIterator();`
`iter.remove();`
`print(aList, iter.nextIndex());`
	`IllegalStateException`

`// 6 pts`
`iter = aList.listIterator();`
`while (iter.hasNext())`
`{`
`  iter.set(iter.next() + iter.previous());`
`  print(aList, iter.nextIndex());`
`  iter.next();`
`}`
	`| AA B C D E`
	`AA | BB C D E`
	`AA BB | CC D E`
	`AA BB CC | DD E`
	`AA BB CC DD | EE`

`// 6 pts`
`iter = aList.listIterator();`
`while (iter.hasNext())`
`{`
`  iter.add(iter.next());`
`  print(aList, iter.nextIndex());`
`}`
	`A A | B C D E`
	`A A B B | C D E`
	`A A B B C C | D E`
	`A A B B C C D D | E`
	`A A B B C C D D E E |`

`// 8 pts`
`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);`
	`E | B C D A`
	`E B C D | A`
	`E D | C B A`
	`E D C | B A`

```
    print(aList, iter.nextIndex());
    print(aList, iter2.nextIndex());
}
```

3a)

```
private void unlink(Node current)
{
      current.previous.next = current.next;
      current.next.previous = current.previous;
}
```

b)

```
private Node findNodeAhead(Node target, int offset)
{
      if (offset < 0)
            throw new IllegalArgumentException("Offset must be non-negative.");

      int count = 0;
      int pureOffset = offset % size;
      Node current = target;

      while (count < pureOffset)
      {
            if (current.next == tail)
                  current = head;
            current = current.next;
            count++;
      }
      return current;
}
```

c)

```
public boolean duplicateGreaterElements(E value, Comparator<? super E> comp)
{
      Node current = head.next;

      // traverse the list
      while (current != tail)
      {
            E curValue = current.data;

            // if encounters an element greater than data, duplicate it.
            if (comp.compare(value, curValue) < 0)
            {
```

```
                Node temp = new Node(curValue);

                // link updates
                temp.previous = current;
                temp.next = current.next;
                current.next.previous = temp;
                current.next = temp;

                current = temp;

                // or, insert temp before current.
                // no need to update current for now.
                //
                // temp.next = current;
                // temp.previous = current.previous;
                // current.previous.next = temp;
                // current.previous = temp;

                ++size;

            }

            current = current.next;
        }


        return true;
    }
```