Com S 228

Fall 2015

Exam 2 Sample Solution

1.

Code snippet	Output	List and iterator state, or exception thrown
`iter = aList.listIterator();`	(none)	`|A B C D E`

`// 3 pts`
`iter = aList.listIterator(aList.size());`
`iter.previous();`
`System.out.println(iter.previousIndex());`
	3	`A B C D | E`

`// 4 pts`
`iter = aList.listIterator();`
`while (iter.hasNext())`
`{`
`  iter.next();`
`  if (iter.hasNext())`
`  System.out.println(iter.next());`
`}`
	B
	D
		`|A B C D E`
		
		After 1st iteration:
		`A B | C D E`
		
		After 2nd iteration:
		`A B C D | E`
		
		After 3rd iteration
		`A B C D E |`
		(answer)

`// 3 pts`
`aList.add("F");`
`aList.remove("C");`
	(none)	`A B D E F`

`// 5 pts`
`aList.add("O");`
`iter = aList.listIterator(aList.size() - 2);`
`iter.previous();`
`iter.set("X");`
`iter.set("Y");`
`iter.add("Z");`
`iter.remove();`
		`IllegalStateException`

`// 5 pts`
`iter = aList.listIterator(aList.size() / 2);`
`while (iter.hasNext())`
`{`
`  iter.next();`
`  iter.previous();`
`  iter.previous();`
`}`
		`NoSuchElementException`

// 8 pts
iter = aList.listIterator();
iter2 = aList.listIterator(aList.size());
while (iter.nextIndex() < iter2.previousIndex()) {
  String s = iter.next();
  String t = iter2.previous();
  if (s.compareTo(t) < 0) {
  iter.set(t);
  iter2.set(s);
  }
}
System.*out*.println(iter.nextIndex() ==
iter2.previousIndex());
iter.next();
iter.add("F");
iter.add("G");
	true	| A B C D E |
		
		During 1st iteration:
		A | B C D | E
		E | B C D | A
		
		During 2nd iteration:
		E B | C | D A
		E D | C | B A
		
		After the while loop:
		E D C | | B A
		E D C F | | B A
		E D C F G | | B A
		(answer)

2a) $O(1)$; b) $O(n)$; c) $O(1)$.

3a) Infix:

2 * b % 3 + (a – (b – (c – d * e))) ^ (4 + b * (c – d)) ^ (1 – 6 / (i – j))

Postfix:

2 b * 3 % a b c d e * – – – 4 b c d - * + 1 6 i j – / – ^ ^ +

b) Postfix:

5 f + 2 a – / b c d e + ^ ^ * 10 2 g h – a b + / * + –

Infix:

(5 + f) / (2 – a) * b ^ c ^ (d + e) – (10 + 2 * ((g – h) / (a + b)))

4. We carry out the following divisions:

748 = 2 * 286 + 176
286 = 1 * 176 + 110
176 = 1 * 110 + 66
110 = 1 * 66 + 44
66 = 1 * 44 + 22
44 = 2 * 22 + 0

Therefore, the GCD is 22.

5a)

```
/**
```

```
     * Compare this shape with a second shape s using their x coordinates, and
     * in the case of a tie, using their y coordinates.
     */
    public int compareTo(Shape s)
    {
        if (x < s.x)
            return -1;
        else if (x > s.x)
            return 1;
        else // x == s.x
        {
            if (y < s.y)
                return -1;
            else if (y > s.y)
                return 1;
            else
                return 0;
        }
    }
```

b)

```
public static <T extends Comparable<? super T>> void selectionSort(T[] arr)
```

c)

```
public class ConvexHull {

    // Data structures
    // ...
    private Point[] pointsToScan;       // points before Graham's scan starts
    private Point[] hullVertices;       // vertices of the constructed convex hull
                                        // in the counterclockwise order
    private PureStack<Point> vertexStack;  // stack used by Graham's scan

    //...

    /**
     * @return  true   if straight movements from p1 to p2 and then from p2 to p3
     *                 make a left turn at p2
     *          false  otherwise
     */
    private boolean leftTurn(Point p1, Point p2, Point p3)
    {
        // implementation details omitted
        // ...
    }
```

```
public void GrahamScan()
{
    // The following are four preprocessing steps that have been implemented
    // already. Do not write code for them.
    //
    // 1. Find the lowest point in the input array of Point objects.  In case
    //    of a tie, pick the leftmost one.
    // ...

    // 2. Remove duplicate points.
    // ...

    // 3. Sort the remaining points in increasing polar angle, and in case of
    //    a tie, in increasing distance to the lowest point.
    // ...

    // 4. Store the sorted sequence in the array pointsToScan[].
    // ...

    vertexStack = new ArrayBasedStack<Point>();

    // Convex hull of one point.
    if (pointsToScan.length == 1)
    {
        // Create hullVertices[] to store the hull.
        // insert code below (2 pts)

        hullVertices = new Point[1];
        hullVertices[0] = pointsToScan[0];

        return;
    }

    // Convex hull of two points.
    if (pointsToScan.length == 2)
    {

        // insert code below (3 pts)

        hullVertices = new Point[2];
        hullVertices[0] = pointsToScan[0];
        hullVertices[1] = pointsToScan[1];


        return;
    }

    // Initialize vertexStack by pushing the first three points
    // from pointsToScan[] onto the stack.

    // insert code below (3 pts)

    vertexStack.push(pointsToScan[0]);
    vertexStack.push(pointsToScan[1]);
```

```
        vertexStack.push(pointsToScan[2]);


        // Scan the remaining points.
        for (int i = 3; i < pointsToScan.length; ++i)
        {

            // Get the next point from pointsToScan[].
            // Fill in the blank in the assignment below.

            Point pi = pointsToScan[i];  // (1 pts)

            while (true)
            {
                Point top = vertexStack.pop();
                Point nextToTop = vertexStack.peek();

                if (leftTurn(nextToTop, top, pi))
                {

                   // insert code below (3 pts)
                    vertexStack.push(top);

                    break;
                }
            }

             // Push the point pi onto the stack.
            // insert code below (2 pts)
            vertexStack.push(pi);
        }


        // Extract the points from vertexStack and store them as vertices
        // in the array hullVertices[], where they will form a counterclockwise
        // traversal of the convex hull as the index increases.


        hullVertices = new Point[vertexStack.size()];
       int i = vertexStack.size() - 1;

        // insert a condition for the while statement below (2 pts)

        while ( !vertexStack.isEmpty() )  // or vertexStack.size() > 0 or i >= 0
        {
             // insert code below (2 pts)
             hullVertices[i--] = vertexStack.pop();
        }
    }

    // other methods of the class ConvexHull
    // ...
}
```