```
import static org.junit.Assert.*;

import java.util.ArrayList;

import org.junit.Test;

/**
 * Project Dijkstra Algorithm
 * This class is used to start the project
 *
 * @author Hakan Tanis
 * @author Kevin Adamczewski
 * @author Jonas Litmeyer
 * Date 30.05.2018
 * @version 3.0
 *
 * Last Change:
 * by: Kevin Adamczewski
 * date: 04.06.2018
 */
public class HalfOrderTest {

	//c1
	@Test
	public void testCheckIfDoubleEdges()
	{
		HalfOrder ho = new HalfOrder();

		ArrayList<Edge> edgelist = new ArrayList<Edge>();
		Node n  = new Node("A", 1, 3);

		edgelist.add(new Edge (8.0));
		edgelist.add(new Edge (10.0));

		//setDestination
		edgelist.get(0).setDestination(n);
		edgelist.get(1).setDestination(n);
		//setSource
		edgelist.get(0).setSource(n);
		edgelist.get(1).setSource(n);

		assertTrue(ho.checkIfDoubleEdges(edgelist, n));
	}

	//c2
	@Test
	public void testCheckIfDoubleEdge1()
	{
		HalfOrder ho = new HalfOrder();

		ArrayList<Edge> edgelist = new ArrayList<Edge>();
```

```
        assertFalse(ho.checkIfDoubleEdges(edgelist,null));
}


// c3-1 e != e2  Dest != Dest    Source != Source
@Test
public void testCheckIfDoubleEdges2()
{
        HalfOrder ho = new HalfOrder();

        ArrayList<Edge> edgelist = new ArrayList<Edge>();
        Node n = new Node("A", 1, 3);
        Node n2 = new Node("B", 2, 4);

        edgelist.add(new Edge (8.0));
        edgelist.add(new Edge (10.0));

        edgelist.get(0).setDestination(n);
        edgelist.get(0).setSource(n2);

        edgelist.get(1).setDestination(n2);
        edgelist.get(1).setSource(n);

        assertFalse(ho.checkIfDoubleEdges(edgelist, n));

}
//c3-2  e!=e2   Dest. != Dest  Source = Source
@Test
public void testCheckIfDoubleEdges3()
{
        HalfOrder ho = new HalfOrder();

        ArrayList<Edge> edgelist = new ArrayList<Edge>();
        Node n = new Node("A", 1, 3);
        Node n2 = new Node("B", 2, 4);

        edgelist.add(new Edge (8.0));
        edgelist.add(new Edge (10.0));

        edgelist.get(0).setDestination(n2);
        edgelist.get(0).setSource(n);

        edgelist.get(1).setDestination(n);
        edgelist.get(1).setSource(n2);

        assertFalse(ho.checkIfDoubleEdges(edgelist, n));

}
//c3-3   e!= e2  Dest == dest  Source != source
@Test
public void testCheckIfDoubleEdges4()
{
```

```
        HalfOrder ho = new HalfOrder();

        ArrayList<Edge> edgelist = new ArrayList<Edge>();
        Node n = new Node("A", 1, 3);
        Node n2 = new Node("B", 2, 4);

        edgelist.add(new Edge (8.0));
        edgelist.add(new Edge (10.0));

        edgelist.get(0).setDestination(n);
        edgelist.get(0).setSource(n);

        edgelist.get(1).setDestination(n);
        edgelist.get(1).setSource(n2);

        assertFalse(ho.checkIfDoubleEdges(edgelist, n));

}
//c3-4 e == e2  d != d  s!=s
@Test
public void testCheckIfDoubleEdges5()
{
        HalfOrder ho = new HalfOrder();

        ArrayList<Edge> edgelist = new ArrayList<Edge>();
        Node n = new Node("A", 1, 3);
        Node n2 = new Node("B", 2, 4);

        edgelist.add(new Edge (8.0));
        edgelist.add(new Edge (10.0));

        edgelist.get(0).setDestination(n2);
        edgelist.get(0).setSource(n);

        edgelist.get(0).setDestination(n);
        edgelist.get(0).setSource(n2);

        assertFalse(ho.checkIfDoubleEdges(edgelist, n));

}
//c3-5 e== e2  d == d  s!= s
@Test
public void testCheckIfDoubleEdges6()
{
        HalfOrder ho = new HalfOrder();

        ArrayList<Edge> edgelist = new ArrayList<Edge>();
        Node n = new Node("A", 1, 3);
        Node n2 = new Node("B", 2, 4);

        edgelist.add(new Edge (8.0));
        edgelist.add(new Edge (10.0));
```

```
        edgelist.get(0).setDestination(n2);
        edgelist.get(0).setSource(n);

        edgelist.get(0).setDestination(n2);
        edgelist.get(0).setSource(n2);

        assertFalse(ho.checkIfDoubleEdges(edgelist, n));

}
// c3-6 e==e2 d==d s==s
@Test
public void testCheckIfDoubleEdges7()
{
        HalfOrder ho = new HalfOrder();

        ArrayList<Edge> edgelist = new ArrayList<Edge>();
        Node n = new Node("A", 1, 3);

        edgelist.add(new Edge (8.0));
        edgelist.add(new Edge (10.0));

        edgelist.get(0).setDestination(n);
        edgelist.get(0).setSource(n);

        edgelist.get(0).setDestination(n);
        edgelist.get(0).setSource(n);

        assertFalse(ho.checkIfDoubleEdges(edgelist, n));

}
// c3-7 e==e d!=d s==s
@Test
public void testCheckIfDoubleEdges8()
{
        HalfOrder ho = new HalfOrder();

        ArrayList<Edge> edgelist = new ArrayList<Edge>();
        Node n = new Node("A", 1, 3);
        Node n2 = new Node("B", 2, 4);

        edgelist.add(new Edge (8.0));

        edgelist.get(0).setDestination(n2);
        edgelist.get(0).setSource(n);

        edgelist.get(0).setDestination(n);
        edgelist.get(0).setSource(n);

        assertFalse(ho.checkIfDoubleEdges(edgelist, n));

}
```

```
	// c3-8 e != e2  d==d s==s
	@Test
	public void testCheckIfDoubleEdges9()
	{
		HalfOrder ho = new HalfOrder();

		ArrayList<Edge> edgelist = new ArrayList<Edge>();
		Node n = new Node("A", 1, 3);
		Node n2 = new Node("B", 2, 4);

		edgelist.add(new Edge (8.0));
		edgelist.add(new Edge (10.0));

		edgelist.get(0).setDestination(n2);
		edgelist.get(0).setSource(n);

		edgelist.get(1).setDestination(n2);
		edgelist.get(1).setSource(n);

		assertTrue(ho.checkIfDoubleEdges(edgelist, n));

	}


}
```