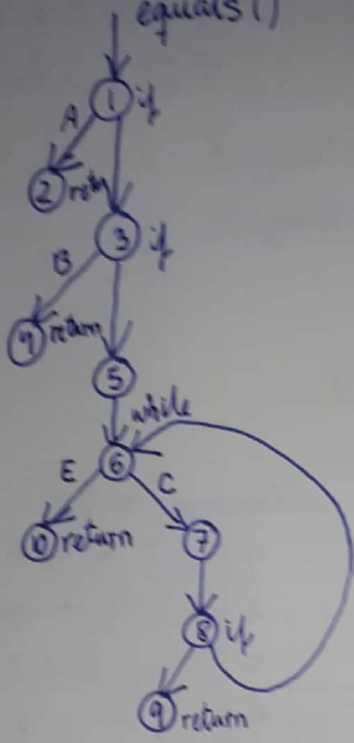
Ejercicio 2

1 y 2



3 Node coverage

RT = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

· Caminos de test:

1) [1, 2]
2) [1, 3, 4]
3) [1, 3, 5, 6, 10]
4) [1, 3, 5, 6, 7, 8, 9]

1) 1er return:

```
List<String> list1 = new ArrayList<String>();
list1.add("foo");
assumeTrue(list1.equals(list1));
```

2) 2º return:

```
List<String> list1 = new ArrayList<String>();
int list2 = 8;
list1.add("foo");
assumeFalse(list1.equals(list2));
```

3) 4º return:

```
List<String> list1 = new ArrayList<String>();
List<String> list2 = new ArrayList<String>();
assumeFalse(list1.equals(list2));
```

4) 3er return:

```
List<String> list1 = new ArrayList<String>();
List<String> list2 = new ArrayList<String>();
list1.add("foo");
list2.add("bar");
assumeFalse(list1.equals(list2));
```

$\boxed{4}$ EP coverage

RT= {(1,2),(1,3),(3,4),(3,5),(5,6),
(6,7),(6,10),(7,8),(8,9),(8,6)}

· Caminos de test:

1) [1,2] → Igual que 1) NC
2) [1,3,4] → Igual que 2) NC
3) [1,3,5,6,7,8,9] → Igual que 4) NC
4) [1,3,5,6,7,8,6,10]

4) 4º return:

```
List <String> list1 = new ArrayList<String>();
List<String> list2= new ArrayList<String>();
list1.add("foo");
list2.add("foo");
assumeTrue (list1.equals(list2));
```

$\boxed{5}$ Prime path

RT = {[1,2],
[1,3,4],
[1,3,5,6,7,8,9],
[6,7,8,6],
[1,3,5,6,10],
[8,6,10],
[8,6,7,8],
[6,7,8,9]}

· Caminos de test:

1) [1,2] → Igual que 1) NC
2) [1,3,4] → Igual que 2) NC
3) [1,3,5,6,7,8,6,7,8,6,7,8,9]
4) [1,3,5,6,7,8,6,10] → Igual que 4) EPC

3) 3er return dando dos vuelta al while:

```
List<String> list1 = new ArrayList<String>();
List<String> list2 = new ArrayList<String>();
list1.add("foo");
list1.add("bar");
list1.add("word");
list2.add("foo");
list2.add("bar");
list2.add("line");
assumeFalse (list1.equals (list2));
```