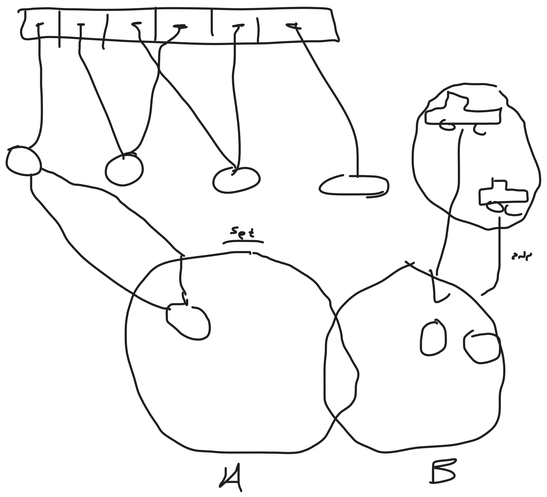
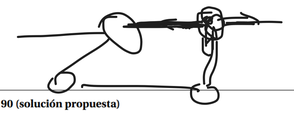
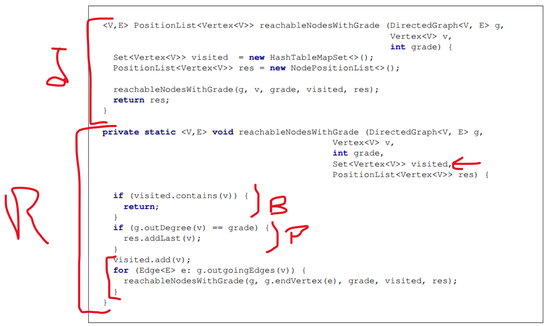




Ejercicio 190 (solución propuesta)

```
static <V,E> Set<Vertex<V>> getVerticesAlcanzables (UndirectedGraph<V, E> g,
                                                    Vertex<V> n) {
  Set<Vertex<V>> visited = new HashTableMapSet<V>();
  getVerticesAlcanzablesRec(g,n,visited);
  return visited;
}
static <V,E> void getVerticesAlcanzablesRec (UndirectedGraph<V, E> g,
                                             Vertex<V> n,
                                             Set<Vertex<V>> visited ) {
  if (visited.contains(n)) {
    return;
  }
  visited.add(n);
  for (Edge<E> e: g.edges(n)) {
    getVerticesAlcanzablesRec(g, g.opposite(n, e), visited);
  }
}
```



```
<V,E> PositionList<Vertex<V>> reachableNodesWithGrade (DirectedGraph<V, E> g,
                                                       Vertex<V> v,
                                                       int grade) {
  Set<Vertex<V>> visited  = new HashTableMapSet<>();
  PositionList<Vertex<V>> res = new NodePositionList<>();

  reachableNodesWithGrade(g, v, grade, visited, res);
  return res;
}
private static <V,E> void reachableNodesWithGrade (DirectedGraph<V, E> g,
                                                   Vertex<V> v,
                                                   int grade,
                                                   Set<Vertex<V>> visited,
                                                   PositionList<Vertex<V>> res) {

  if (visited.contains(v)) {
    return;
  }
  if (g.outDegree(v) == grade) {
    res.addLast(v);
  }
  visited.add(v);
  for (Edge<E> e: g.outgoingEdges(v)) {
    reachableNodesWithGrade(g, g.endVertex(e), grade, visited, res);
  }
}
```