1. Let $G = (V, E)$ be an undirected graph. Modify the following algorithm to determine if $G$ is a tree. (A tree is a connected graph without cycles.) Your modification must not increase the time complexity of the algorithm. Do NOT rewrite the whole algorithm; just make the necessary changes.

```
procedure ConnectedComponents($G = (V, E)$)
    // Assume $V = \{1, 2, \ldots, n\}$
    // global array component[1..n]
    component[1..n] ← 0
    cn ← 0
    for $i ← 1$ to $n$
        if component[$i$] = 0 then
            cn ← cn + 1
            call dfs($i, cn$)

procedure dfs($v, cn$)
    component[$v$] ← cn;
    for each node $w$ such that $(v, w) ∈ E$ do
        if component[$w$] = 0 then call dfs($w, cn$)
```
2. Let $G = (V, E)$ be a directed graph. Modify the following algorithm to print all back edges. Your modification must not increase the time complexity of the algorithm. Do NOT rewrite the whole algorithm; just make the necessary changes.

```plaintext
procedure Search($G = (V, E)$)
    // Assume $V = \{1, 2, \ldots, n\}$ //
    // time, $vn[1..n]$, and $fn[v]$ are global variables //
    time ← 0;
    $vn[1..n] ← 0$;
    $fn[1..n] ← 0$;
    for $i ← 1$ to $n$
        if $vn[i] = 0$ then call $dfs(i)$

procedure $dfs(v)$
    $vn[v] ← time ← time + 1$;
    for each node $w$ such that $(v, w) \in E$ do
        if $vn[w] = 0$ then call $dfs(w)$;
    $fn[v] ← time ← time + 1$
```