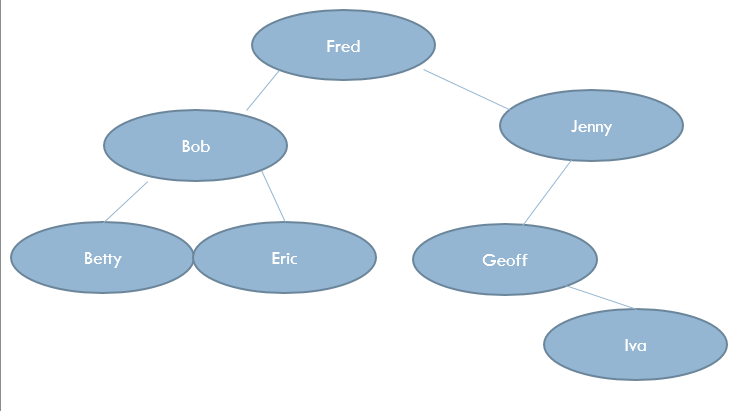
**Post-Order Tree Traversal Algorithm**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Procedure Call | P | Tree[p].left | Tree[p].right | Output |
| P1 |  |  |  |  |
| P2 |  |  |  |  |
| P3 |  |  |  |  |
| P2 |  |  |  |  |
| P4 |  |  |  |  |
| P2 |  |  |  |  |
| P1 |  |  |  |  |
| P5 |  |  |  |  |
| P6 |  |  |  |  |
| P7 |  |  |  |  |
| P6 |  |  |  |  |
| P5 |  |  |  |  |
| P1 |  |  |  |  |

Procedure calls are complete for you to help you return correctly to the previous invocation.

|  |  |  |  |
| --- | --- | --- | --- |
| Position | Lptr | Data | Rptr |
| 1 | 3 | Fred | 2 |
| 2 | 5 | Jenny | 0 |
| 3 | 6 | Bob | 4 |
| 4 | 0 | Eric | 0 |
| 5 | 0 | Geoff | 7 |
| 6 | 0 | Betty | 0 |
| 7 | 0 | Iva | 0 |

1. **Complete the table for the given Binary Tree**
2. **Complete the trace table using Post-Order Algorithm starting with traverse\_from(1)**