**In-Order Tree Traversal Algorithm**

Procedure traverse\_from(p)

if tree[p].left <> 0 then

traverse\_from(left)

endif

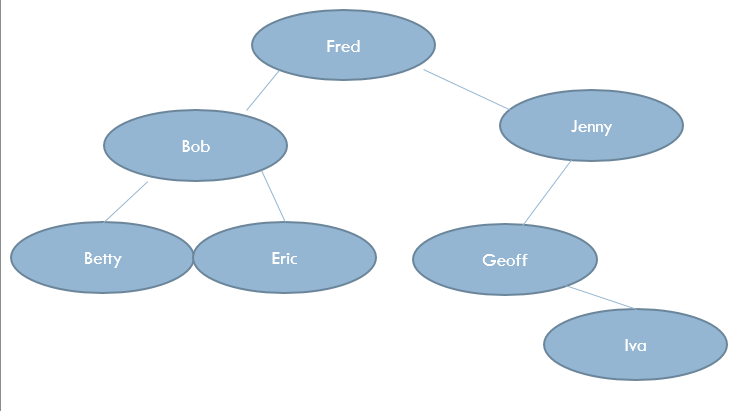
Output data

If tree[p].right <> 0 then

traverse\_from(right)

endif

End proc



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Procedure Call | P | Tree[p].left | Tree[p].right | Output |
| P1 | 1 | 3 |  |  |
| P2 | 3 | 6 |  |  |
| P3 | 6 | 0 | 0 | Betty |
| P2 | 3 | 6 |  |  |
| 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 |  | Fred |  |
| 2 |  | Jenny |  |
| 3 |  | Bob |  |
| 4 |  | Eric |  |
| 5 |  | Geoff |  |
| 6 |  | Betty |  |
| 7 |  | Iva |  |

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