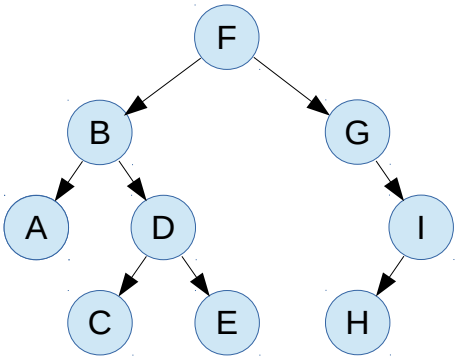
1 Binary Tree Traversal

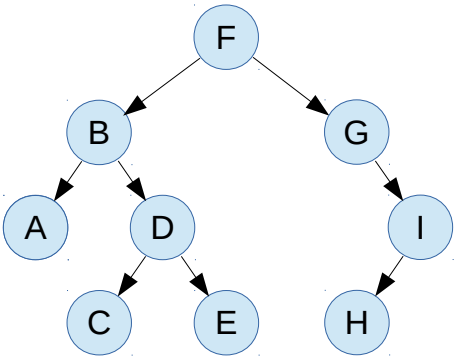
다음에 주어진 이진트리에 대하여 recursive 알고리즘과 iterative 알고리즘을 simulation하는 문제이다. 빈칸 (node와 stack)을 채우시오.

```
inorder(node)
  if (node = null)  return
  inorder(node.left)
  visit(node)
  inorder(node.right)
```



inorder(F) if (F = null) return **inorder**(B) print(F) **inorder**(G)	**inorder**(B) if (B = null) return **inorder**(A) print(B) **inorder**(D)	**inorder**(A) if (A = null) return **inorder**(null) print(A) **inorder**(null)
inorder(D) if (D = null) return **inorder**(C) print(D) **inorder**(E)	**inorder**(C) if (C = null) return **inorder**(null) print(C) **inorder**(null)	**inorder**(E) if (E = null) return **inorder**(null) print(E) **inorder**(null)
inorder(G) if (G = null) return **inorder**(null) print(G) **inorder**(I)	**inorder**(I) if (I = null) return **inorder**(H) print(I) **inorder**(null)	**inorder**(H) if (H = null) return **inorder**(null) print(H) **inorder**(null)

```
iterativeInorder(node)
  s ← empty stack

  while (not s.isEmpty()
         or node ≠ null)
    if (node ≠ null)
      s.push(node)
      node ← node.left
    else
      node ← s.pop()
      visit(node)
      node ← node.right
```




```
while (...)
  if (F ≠ null)
    s.push(F)
    node ← B

Stack:F
```

```
while (...)
  if (B ≠ null)
    s.push(B)
    node ← A

Stack:FB
```

```
while (...)
  if (A ≠ null)
    s.push(A)
    node ← null

Stack:FBA
```

```
while (...)
  else
    node ← s.pop
    print(A)
    node ←null

Stack:FB
```

```
while (...)
  else
    node ← s.pop
    print(B)
    node ←D

Stack:F
```

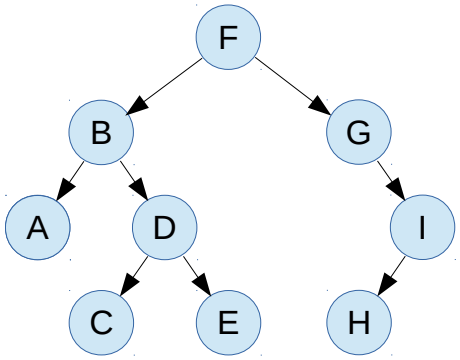
```
while (...)
  if (D ≠ null)
    s.push(D)
    node ← C

Stack:FD
```

```
while (...)
  if (C ≠ null)
    s.push(C)
    node ← null

Stack:FDC
```

```
while (...)
  else
    node ← s.pop
    print(C)
    node ←null

Stack:FD
```

```
while (...)
  else
    node ← s.pop
    print(D)
    node ←E

Stack:F
```

```
iterativeInorder(node)
 s ← empty stack

 while (not s.isEmpty()
          or node ≠ null)
  if (node ≠ null)
    s.push(node)
    node ← node.left
  else
    node ← s.pop()
    visit(node)
    node ← node.right
```



while (...) if (E ≠ null) s.**push**(E) node ← null **Stack**:FE	**while** (...) else node ← s.**pop** print(E) node ←null **Stack**:F	**while** (...) else node ← s.**pop** print(F) node ←G **Stack**:
while (...) if (G ≠ null) s.**push**(G) node ← null **Stack**:G	**while** (...) else node ← s.**pop** print(G) node ←I **Stack**:	**while** (...) if (I ≠ null) s.**push**(I) node ← H **Stack**:I
while (...) if (H ≠ null) s.**push**(H) node ← null **Stack**:IH	**while** (...) else node ← s.**pop** print(H) node ←null **Stack**:I	**while** (...) else node ← s.**pop** print(I) node ←null **Stack**: