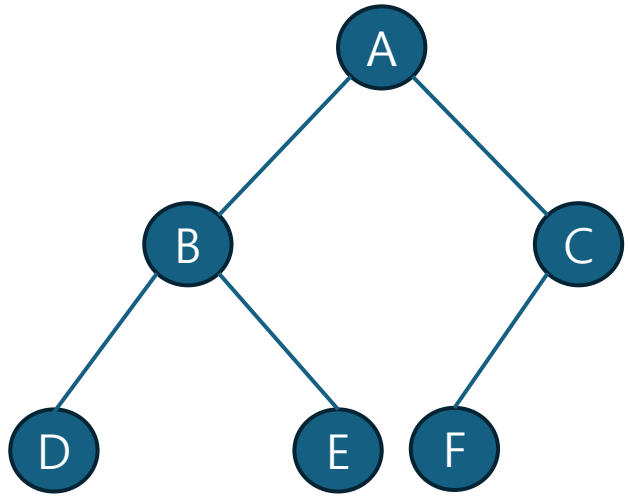
6 Week Practice

[Tree]

[Code] main.py CircularQueue.py, BinaryNode.py, BinaryTree.py

[Basic 1] Tree 구조 이해

- getCount_: 이진 트리의 노드 개수 세는 함수
- getRightCount_: 오른쪽 자식 노드를 가지는 노드들의 개수
- getLeafCount_: 이진 트리의 leaf node의 개수
- getHeight_: 루트 노드에서 가장 깊은 리프 노드까지의 경로에 있는 노드 수
- calcSize_: 이진 트리에서 모든 노드의 데이터 값을 합산하는 것




```
노드의 개수 = 6
오른쪽 자식 노드를 가지는 노드의 개수 = 2
단말의 개수 = 3
트리의 높이 = 3

inorder: [D][B][E][A][F][C]
preorder: [A][B][D][E][C][F]
postorder: [D][E][B][F][C][A]
levelorder: [A][B][C][D][E][F]
각 수를 제곱해서 주어진 기호대로 사칙연산 했을 때의 계산 결과 = 25
디렉토리 용량 계산 결과 = 850 KB
```

[Code] main.py CircularQueue.py, BinaryNode.py, BinaryTree.py, BinSrchTree.py

[Basic 2] Binary tree 구현

- Search: 해당 트리에서 해당 key를 갖고있는 노드 찾기
- search_: 찾고 있는 key인지 판단
- Insert_, remove_ 동일

```
inorder: [3][7][12][18][22][26][30][35][68][99]
preorder: [35][18][7][3][12][26][22][30][68][99]
postorder: [3][12][7][22][30][26][18][99][68][35]
levelorder: [35][18][68][7][26][99][3][12][22][30]
노드의 개수 = 10
단말의 개수 = 5
트리의 높이 = 4
탐색 성공: 키값이 26인 노드 있음
탐색 실패: 키값이 25인 노드 없음
삭제: case 1 ==> 노드 3 삭제
levelorder: [35][18][68][7][26][99][12][22][30]
삭제: case 2 ==> 노드 68 삭제
levelorder: [35][18][99][7][26][12][22][30]
삭제: case 3 ==> 노드 18 삭제
levelorder: [35][22][99][7][26][12][30]
삭제: case 3 ==> 노드 35 삭제 (루트 노드 삭제)
levelorder: [99][22][7][26][12][30]
노드의 개수 = 6
단말의 개수 = 2
트리의 높이 = 4
```

[Code] main.py, BinaryNode.py, BinaryTree.py

[Advanced]

- CheckLevel: 몇 번째 레벨(level)에 위치하는 지
- checkBalanced: 균형잡힌 이진 트리인지 확인
- calcPathLength: 트리의 전체 경로 길이(Path Length)를 계산
- swapNodes: 트리의 전체 노드 좌우반전
- hasSameNode: 노드가 서브트리에 포함되어 있는지
- checkValid: 유효한 트리인지 확인

```
노드 개수 = 6
단말 개수 = 3
트리 높이 = 3
preorder: [A] [B] [C] [D] [E] [F]


(1)
완전이진트리가 아닙니다

(2)
노드의 레벨은 1.
노드의 레벨은 2.
노드의 레벨은 3.
노드의 레벨은 3.
노드의 레벨은 2.
노드의 레벨은 3.

(3)
균형잡힌 이진트리입니다.

(4)
전체 경로의 길이는 8입니다.

(5)
트리의 좌우를 교환합니다.
preorder: [A] [E] [F] [B] [D] [C]

(6)
루트b인 트리와 루트e인 트리는 Disjoint합니다.

(7)
Valid한 이진트리입니다.
```