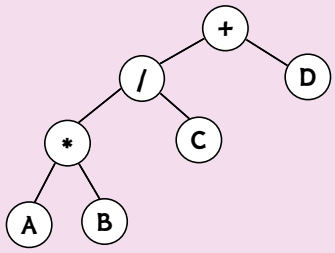
5. 트리 운행

(1) 이진트리 운행

다음은 이항연산자로 구성된 수식 'A * B / C + D'를 트리로 나타낸 것이다.

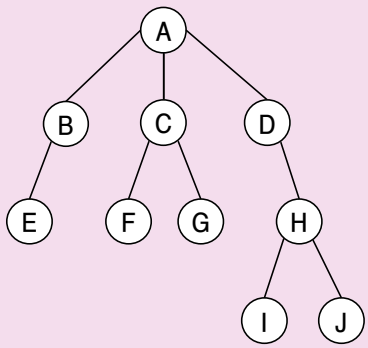


- **전위운행(중간,좌,우)** : + / * A B C D
- **중위운행(좌,중간,우)** : A * B / C + D
- **후위운행(좌,우,중간)** : A B * C / D +

• 이진트리 운행은 **수식**의 Prefix, Infix, Postfix와 자연스럽게 **대응**된다.

전위운행 (preorder traversal)	먼저 부모노드를 방문하고 왼쪽으로 가면서 계속 방문한다. 더 이상 왼쪽이 없으면 오른쪽으로 이동하면서 앞의 원리와 같이 방문되지 않은 노드를 방문한다.
중위운행 (inorder traversal)	왼쪽으로 더 이상 갈 수 없을 때까지 이동해서 그 노드를 방문하고 부모노드, 오른쪽 자식노드를 방문한다.
후위운행 (postorder traversal)	왼쪽으로 더 이상 갈 수 없을 때까지 이동해서 그 노드를 방문하고 오른쪽 자식노드, 부모노드 순으로 방문한다.

(2) 일반트리 운행



① Level Order 운행법

- 하향식 : 근노드→단말노드(A B C D E F G H I J)
- 상향식 : 단말노드→근노드(I J E F G H B C D A)

② Familly 운행법 : A B C D H I J F G E

먼저 근노드를 검사한 후 근노드의 자식노드를 왼쪽에서 오른쪽 순으로 검사한 다음, 가장 늦게 검사된 자식노드부터 앞의 작업을 반복한다.

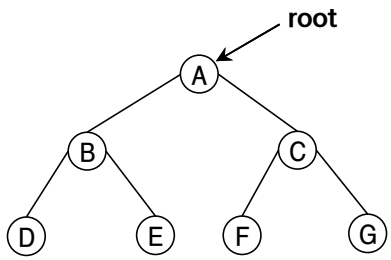
③ Preorder 운행법 : A B E C F G D H I J

④ Postorder 운행법 : E B F G C I J H D A

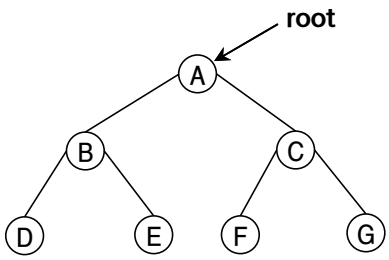
(3) 이진트리 운행 알고리즘

- **출력문 위치**에 의해 **전위운행, 중위운행, 후위운행**이 결정된다.
- **재귀호출**을 기준으로 **출력문**이 앞쪽에 있으면 **전위운행**이다.
- **재귀호출**을 기준으로 **출력문**이 중간에 있으면 **중위운행**이다.
- **재귀호출**을 기준으로 **출력문**이 뒤쪽에 있으면 **후위운행**이다.

```
typedef struct Node                 // 노드구조 정의
{
  struct Node *left;                // 좌측 자식 포인터
  char data;                        // 데이터
  struct Node *right;               // 우측 자식 포인터
} Node;

void preorder(Node *node)           // 전위운행
{
  if( !node ) return;
  printf(" %c", node->data);        // 전위운행 데이터 출력
  preorder(node->left);
  preorder(node->right);
}

void inorder(Node *node)            // 중위운행
{
  if( !node ) return;
  inorder(node->left);
  printf(" %c", node->data);        // 중위운행 데이터 출력
  inorder(node->right);
}

void postorder(Node *node)          // 후위운행
{
  if( !node ) return;
  postorder(node->left);
  postorder(node->right);
  printf(" %c", node->data);        // 후위운행 데이터 출력
}
```



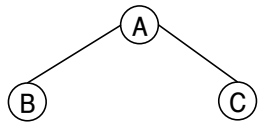
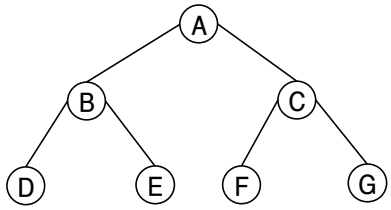
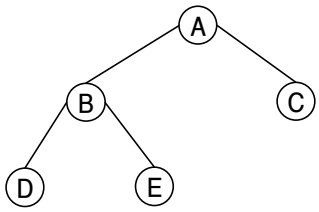
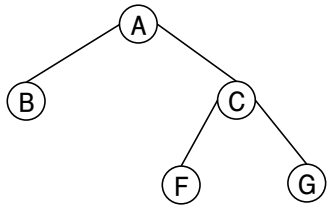
```
Node *newNode(int data)              // 새로운 노드 추가
{
    Node *node = (Node *)malloc(sizeof(Node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return(node);
}

void main()
{
  Node *root = newNode('A');         // 루트노드 A 생성
  root->left = newNode('B');         // 좌측 자식노드 생성
  root->right = newNode('C');        // 우측 자식노드 생성
  root->left->left = newNode('D');
  root->left->right = newNode('E');
  root->right->left = newNode('F');
  root->right->right = newNode('G');
  printf("\n전위운행 : "); preorder(root);
  printf("\n중위운행 : "); inorder(root);
  printf("\n후위운행 : "); postorder(root);
}
```




```
전위운행 :  A B D E C F G
중위운행 :  D B E A F C G
후위운행 :  D E B F G C A
```

(4) 이진트리 혼합운행

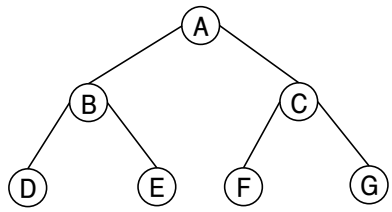
이진트리 **혼합운행**은 **전위운행, 중위운행, 후위운행**이 혼합되어 있는 구조이다.

<table>
<tr><td> 
</td><td>전위(중,좌,우) : A B C
중위(좌,중,우) : B A C
후위(좌,우,중) : B C A
↓혼합운행
전위 + 중위 : A B B A C C
전위 + 후위 : A B B C C A
중위 + 후위 : B B A C C A
전위 + 중위 + 후위 : A B B B A C C C A</td></tr>
<tr><td> 
</td><td>전위(중,좌,우) : A B D E C F G
중위(좌,중,우) : D B E A F C G
후위(좌,우,중) : D E B F G C A
↓혼합운행
전위 + 중위 : A B D D B E E A C F F C G G
전위 + 후위 : A B D D E E B C F F G G C A
중위 + 후위 : D D B E E B A F F C G G C A</td></tr>
<tr><td colspan="2">전위 + 중위 + 후위 : A B D D D B E E E B A C F F F C G G G C A</td></tr>
<tr><td> 
</td><td>전위 + 중위 : A B D D B E E A C C
전위 + 후위 : A B D D E E B C C A
중위 + 후위 : D D B E E B A C C A</td></tr>
<tr><td> 
</td><td>전위 + 중위 : A B B A C F F C G G
전위 + 후위 : A B B C F F G G C A
중위 + 후위 : B B A F F C G G C A</td></tr>
</table>

- **이진트리 운행**은 **전위운행, 중위운행, 후위운행**을 **혼합**하여 운행할 수 있다.

- **출력문 위치**에 의해 **전위운행, 중위운행, 후위운행**이 결정된다.
- **재귀호출**을 기준으로 **출력문**이 앞쪽에 있으면 **전위운행**이다.
- **재귀호출**을 기준으로 **출력문**이 중간에 있으면 **중위운행**이다.
- **재귀호출**을 기준으로 **출력문**이 뒤쪽에 있으면 **후위운행**이다.

예제 : 〈보기 1〉의 트리를 〈보기 2〉처럼 수행한 결과는?

<table>
<tr><th>보기 1</th><th>보기 2</th></tr>
<tr><td> 
</td><td>

```
typedef struct Node{
  char data;                  // 노드에 저장된 문자
  struct Node *left, *right;  // 자식 포인터
} Node;
void traverse(Node *node){
  if( !node ) return;
  printf(" %c", node->data);  // 전위운행 데이터 출력
  traverse(node->left);
  printf(" %c", node->data);  // 중위운행 데이터 출력
  traverse(node->right);
}
```

</td></tr>
<tr><td colspan="2">· 출력문 위치에 의해 전위운행, 중위운행, 후위운행이 결정된다.
· 재귀호출을 기준으로 출력문이 앞쪽에 있으면 전위운행이다.
· 재귀호출을 기준으로 출력문이 중간에 있으면 중위운행이다.
· 재귀호출을 기준으로 출력문이 뒤쪽에 있으면 후위운행이다.</td></tr>
</table>

↓

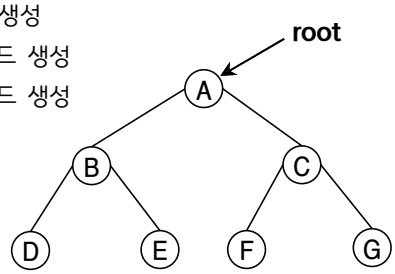
↓ **이진트리 운행 : 전위운행 + 중위운행**

↓ **완성된 프로그램으로 구현하면 다음과 같다.**

↓

```
// 이진트리 운행 : 전위운행 + 중위운행
#include<stdio.h>
typedef struct Node             // 이진트리 노드구조
{
  char data;                    // 노드에 저장된 문자
  struct Node *left, *right;    // 자식 포인터
} Node;

Node *newNode(int data)         // 새로운 노드 추가
{
    Node *node = (Node *)malloc(sizeof(Node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return(node);
}
```

```
void traverse(Node *node)            // 이진트리 운행
{
  if( !node ) return;                // 공트리이면 종료
  printf(" %c", node->data);         // 전위운행으로 데이터 출력
  traverse(node->left);              // 좌측으로 이동
  printf(" %c", node->data);         // 중위운행으로 데이터 출력
  traverse(node->right);             // 우측으로 이동
}

void main()
{
  Node *root = newNode('A');         // 루트노드 A 생성
  root->left = newNode('B');         // 좌측 자식노드 생성
  root->right = newNode('C');        // 우측 자식노드 생성
  root->left->left = newNode('D');
  root->left->right = newNode('E');
  root->right->left = newNode('F');
  root->right->right = newNode('G');
  printf("\n방문되는 노드 순서 : ");
  traverse(root);                    // 방문순서 출력
}
```



// 문제 분석

- 주어진 문제는 **전위운행**과 **중위운행**이 **혼합**되어 있는 구조이다.
- **전위(중,좌,우)** : A B D E C F G
- **중위(좌,중,우)** : D B E A F C G

↓

↓ **전위**운행과 **중위**운행이 **혼합**되어 출력

↓

- **전위 + 중위** : A B D D B E E A C F F C G G

↓ ↓

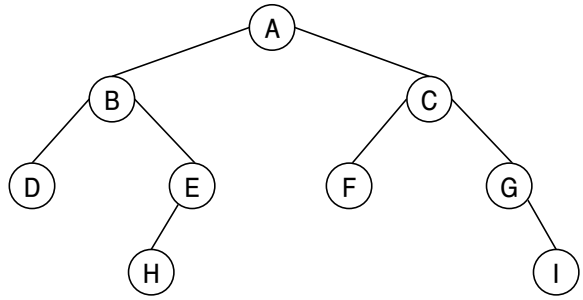
↓ **중위운행**에서 루트 A는 **중간**에 방문

↓

전위운행에서 루트 A는 **처음**에 방문

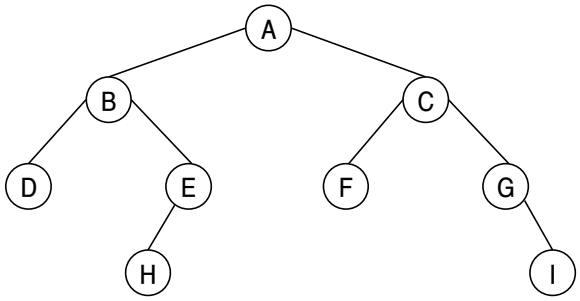
기출문제 분석

1. 다음 이진트리에 대하여 전위순회(preorder traversal), 중위순회 (inorder traversal), 후위순회 (postorder traversal)를 수행하였다. 3가지 순회 방법 중 2가지 이상에서 방문 순서가 동일한 노드 수는? (예를 들어, 3개의 노드 A, B, C를 갖는 이진트리에서 전위순회 : B-A-C, 중위순회 : A-B-C, 후위순회 : A-C-B의 결과를 얻었다면, 중위순회 및 후위순회에서 노드 A에 대한 방문 순서가 동일하고 전위순회 및 중위순회에서 노드 C에 대한 방문 순서가 동일하므로, 2가지 이상에서 방문 순서가 동일한 노드 수는 2개이다) [2016년 국가 7급]



① 2 ② 3 ③ 4 ④ 5

♤ 이진트리 순회



	1	2	3	4	5	6	7	8	9
전위순회(중간, 좌, 우)	A	**B**	D	**E**	H	C	F	**G**	**I**
중위순회(좌, 중간, 우)	**D**	**B**	H	**E**	A	F	C	**G**	**I**
후위순회(좌, 우, 중간)	**D**	H	E	B	F	I	G	C	A

- 2가지 이상에서 방문 순서가 동일한 경우는 전부 5개이다.(진한 부분)
- 결과를 알면 매우 쉽지만, 처음 문제를 읽으면 무슨 말인지? 문제가 참 그렇다.

정답 : ④

2. 다음은 어떤 이진트리에 대한 전위(pre-order) 및 중위(in-order) 순회 시 방문순서이다. 이 이진트리에 대한 설명으로 가장 옳지 않은 것은? [2022년 군무원 7급]

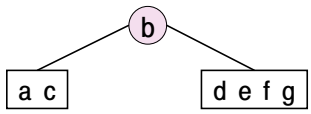
전위순회 : b a c f d e g
중위순회 : a c b d e f g
---

① 루트노드는 b이다.
② 단말노드 중에는 c와 e가 있다.
③ 후위(post-order) 순회 시 마지막 바로 전에 방문노드는 g이다.
④ 후위(post-order) 순회 시 처음 방문노드는 c이다.

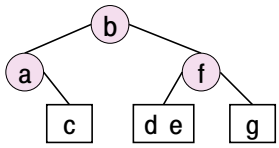
♤ **이진트리 순회**

• 전위순회(중,좌,우) : **b** a c f d e g ← **b**가 근노드이다.
• 중위순회(좌,중,우) : a c **b** d e f g ← **b**를 중심으로 좌우측에 배치

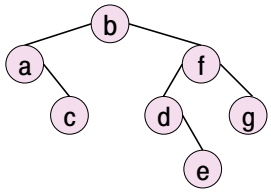
↓전위순회는 근노드를 가장 **먼저** 방문한다. **b**가 근노드라는 것을 알 수 있다.
↓중위순회는 근노드를 **중간**에 방문한다. 근노드 **b**를 중심으로 좌우측에 배치된다.



↓전위순회를 참조하면, 서브트리에서 **a와 f**가 각각 근노드라는 것을 알 수 있다.
↓중위순회를 참조하면, 서브트리에서 **a**의 우측에 **c**, **f**의 좌측에 **d e** 우측에 **g**이다.



↓같은 원리로, 서브트리에서 **d**가 근노드이고 **d**의 우측에 **e**이다.



후위순회(좌,우,중) : **c a e d g f b**

• 후위순회 시 마지막 바로 전에 방문노드는 **f**이다.

정답 : ③

3. 키 값이 1, 2, 3, 4, 5, 6, 7, 8인 8개의 노드로 구성된 이진탐색트리를 전위순회(preorder traversal)하면, 키 값 6, 2, 1, 4, 3, 5, 8, 7 순서로 노드를 방문한다. 이 트리를 후위순회(postorder traversal)할 경우 방문 노드의 키 값을 방문 순서대로 바르게 나열한 것은? [2020년 국가 7급]

① 1, 2, 3, 4, 5, 6, 7, 8　　② 1, 3, 2, 6, 5, 8, 7, 4

③ 1, 3, 5, 4, 2, 7, 8, 6　　④ 8, 7, 6, 5, 4, 3, 2, 1

♤ **이진탐색트리 순회**

• 먼저, 이진탐색트리를 **중위순회** 하면 **오름차순으로 정렬**된다.

↓

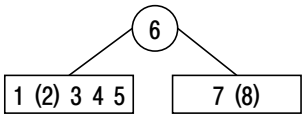
• 전위순회(중, 좌, 우) : **(6)**, 2, 1, 4, 3, 5, 8, 7 ← 6이 근노드이다.

• 중위순회(좌, 중, 우) : 1, 2, 3, 4, 5, **(6)**, 7, 8 → 6을 기준으로 좌우측으로 배치

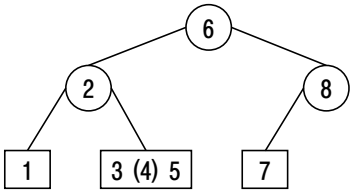
• 후위순회(좌, 우, 중) : ?

↓ 전위순회는 근노드를 가장 먼저 방문한다. **6**이 근노드라는 것을 알 수 있다.

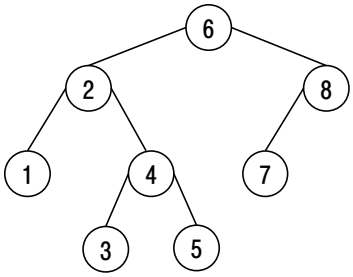
↓ 중위순회는 근노드를 중간에 방문한다. 근노드를 중심으로 좌우측에 배치된다.



↓ 서브트리에서 2와 8이 근노드이므로



↓ 서브트리에서 4가 근노드이므로



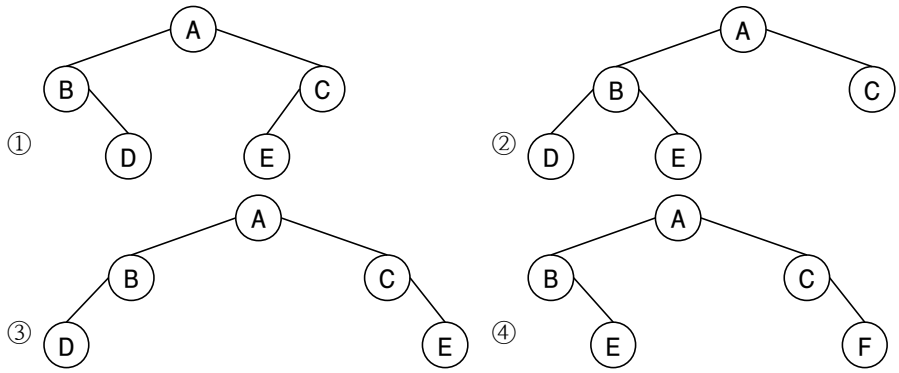
완성된 이진탐색트리

• 후위순회(좌, 우, 중) : **1, 3, 5, 4, 2, 7, 8, 6**

정답 : ③

4. <보기>는 5개의 노드로 구성된 이진트리를 전위, 중위, 후위 순회한 결과이다. 이 결과로 알 수 있는 이진트리의 구성은? [2020년 서울 7급]

----<보기>----------
- 전위순회 : A - B - D - C - E
- 중위순회 : B - D - A - E - C
- 후위순회 : D - B - E - C - A



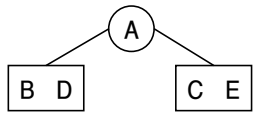
♤ **이진트리를 전위, 중위, 후위 순회**

- 전위순회(중, 좌, 우) : A - B - D - C - E ← A가 근노드이다.
- 중위순회(좌, 중, 우) : B - D - A - E - C ← A을 중심으로 좌우측으로 배치
- 후위순회(좌, 우, 중) : D - B - E - C - A ← A가 근노드이다.

↓

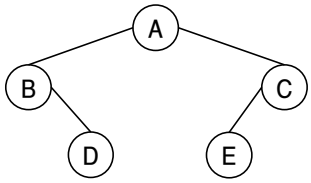
↓ 전위순회는 근노드를 가장 **먼저** 방문한다. A가 근노드라는 것을 알 수 있다.

↓ 중위순회는 근노드를 **중간**에 방문한다. 근노드 A를 중심으로 좌우측에 배치된다.



↓ 전위순회를 참조하면, 서브트리에서 B와 C가 각각 근노드라는 것을 알 수 있다.

↓ 중위순회를 참조하면, 서브트리에서 D는 B의 우측, E는 C의 우측임을 알 수 있다.



완성된 이진탐색트리

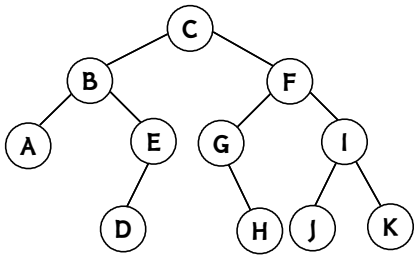
정답 : ①

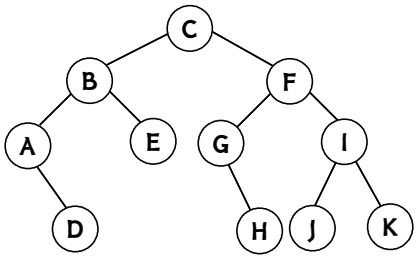
5. 다음은 이진트리를 후위순회와 중위순회로 방문한 결과이다. 이 두 가지 순회 결과를 이용하여 이진트리를 구성한 것으로 옳은 것은? [2015년 국가 7급]

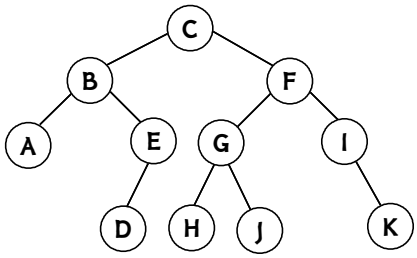
후위순회 : A D E B H G J K I F C

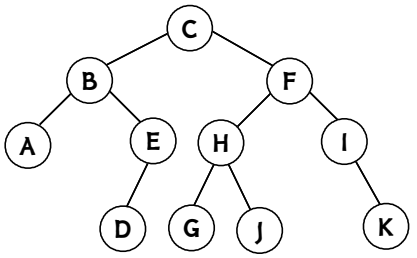
중위순회 : A B D E C G H F J I K

---

① 

② 

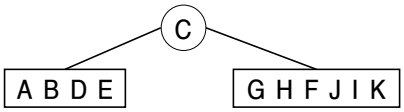
③ 

④ 

♤ 이진트리 순회(후위순회 : 좌→우→중간, 중위순회 : 좌→중간→우)

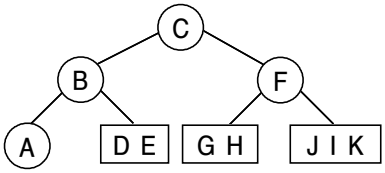
㉠ 후위순회는 근노드를 가장 나중에 방문한다. C가 근노드라는 것을 알 수 있다.
중위순회는 근노드를 중간에 방문한다. 근노드를 중심으로 좌우측에 배치된다.

- 후위순회 : A D E B H G J K I F (C) ← C가 근노드이다.
- 중위순회 : A B D E (C) G H F J I K → C를 기준으로 좌우측에 배치



㉡ 근노드 C를 제외한 나머지 부분에서

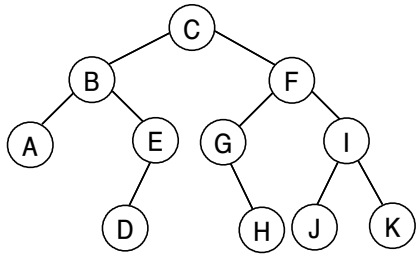
- 후위순회 : A D E (B) H G J K I (F) ← 좌우 서브트리에서 B, F가 각각 근노드
- 중위순회 : A (B) D E G H (F) J I K → B, F를 기준으로 각각 좌우측에 배치



→ 하나만 있는 것은 위치가 확정된 것이다.(A)

㉣ 설명한 원리대로 다음 단계를 거치면, 완성된 트리를 얻을 수 있다.

- 후위순회 : D (E) H (G) J K (I) ← 서브트리에서 (E) (G) (I)가 각각 근노드
- 중위순회 : D (E) (G) H J (I) K → (E) (G) (I)를 기준으로 각각 좌우측에 배치

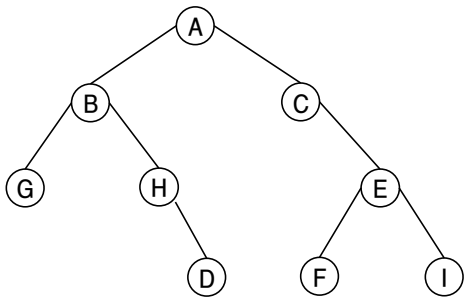


// 이진트리 순회 결과를 가지고 원래의 이진트리를 찾는 문제

- (전위, 중위) 또는 (중위, 후위) 순회 쌍의 운행 결과가 있어야 원래의 이진트리를 찾을 수 있다.
- **(전위순회, 후위순회)** 쌍의 운행 결과만으로는 원래의 이진트리를 찾을 수 **없다.**

정답 : ①

6. 다음 이진트리(binary tree)의 순회 방법으로 옳지 않은 것은? [2017년 국가 7급]



① 전위(preorder) 순회 : A B G H D C E F I
② 중위(inorder) 순회 : G B H D A C F E I
③ 후위(postorder) 순회 : G D H B F E I C A
④ 레벨 순서(level order) 순회 : A B C G H E D F I

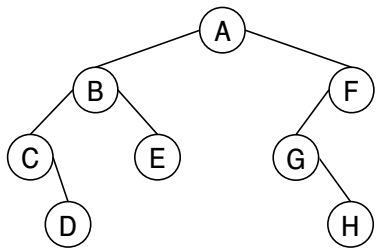
♤ **이진트리 순회**

- 후위순회(좌,우,중) : G D H B F I E C A

정답 : ③

7. 노드의 필드가 *lchild, data, *rchild인 연결리스트로 구현된 다음 그림의 이진트리를 함수 mystery_Order()에 의해 운행할 때 방문되는 노드 순서가 맞는 것은? (단, 초기 호출시 t는 루트인 노드 A를 가리킨다) [2021년 군무원 7급]

```
void mystery_Order (treePointer t)
{
  if (t)
  {
    mystery_Order(t->lchild);
    printf("%c", t->data);
    mystery_Order(t->rchild);
    printf("%c", t->data);
  }
}
```

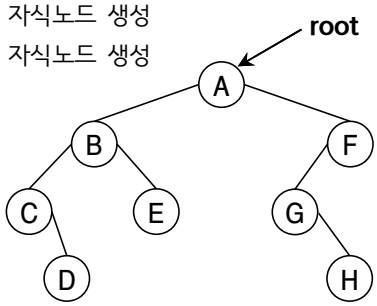


① C D D C B E E B A G H H G F F A
② A B C D D C E E B A F G H H G F
③ A B C D E F G H D C E B H G F A
④ C C D D E E B B A A G G H H F F

♤ 이진트리 운행 : 중위운행(좌,중,우) + 후위운행(좌,우,중)

```
// 방문되는 노드 순서에 대한 완성된 프로그램
#include<stdio.h>
struct node                          // 이진트리 노드구조
{
  char data;
  struct node *left;
  struct node *right;
};
struct node *newNode(int data)    // 새로운 노드 추가
{
    struct node *node = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return(node);
}
```

```
void mystery_Order(struct node *root)    // 이진트리 운행
{
  if(root)                               // 공백 이진트리가 아니면
  {
    mystery_Order(root->left);           // 좌측으로 이동
    printf("% c", root->data);           // 중위운행으로 데이터 출력
    mystery_Order(root->right);          // 우측측으로 이동
    printf("% c", root->data);           // 후위운행으로 데이터 출력
  }
}

void main()
{
  struct node *root = newNode('A');      // 루트노드 A 생성
  root->left = newNode('B');             // 좌측 자식노드 생성
  root->right = newNode('F');            // 우측 자식노드 생성
  root->left->left = newNode('C');
  root->left->right = newNode('E');
  root->right->left = newNode('G');
  root->left->left->right = newNode('D');
  root->right->left->right = newNode('H');
  printf("\n방문되는 노드 순서 : ");
  mystery_Order(root);                   // 방문 순서 출력
}
```



// 문제 분석

- 주어진 문제는 **중위운행**과 **후위운행**이 **혼합**되어 있는 구조이다.**(이런 문제는 처음 봄)**
- **중위운행(좌,중,우)** : C D B E A G H F
- **후위운행(좌,우,중)** : D C E B H G F A

↓

↓ **중위**운행과 **후위**운행이 혼합되어 출력

↓

- **방문 순서** : C D D C B E E B A G H H G F F A

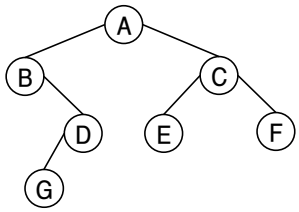
↓ ↳ **후위운행**에서 루트 A는 **마지막**에 방문

중위운행에서 루트 A는 **중간**에 방문

정답 : ①

8. 〈보기 1〉의 트리의 루트(root)가 A 노드일 때, 루트노드 A를 인자로 하여 〈보기 2〉의 함수를 수행한 결과는? [2021년 서울 7급]

보기 1	보기 2

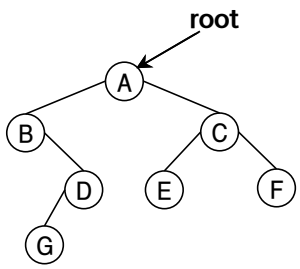
	typedef struct Node{
	  char data; // 노드에 저장된 문자
	  struct Node *left, *right; // 자식 포인터
	} Node;
	void traverse(Node *node){
	  if(!node) return;
	  traverse(node->left);
	  printf(" %c", node->data);
	  traverse(node->right);
	  printf(" %c", node->data);
	}

① A B D G C E F G D B E F C A
② A B D G G D B C E E F F C A
③ B G D A E C F G D B E F C A
④ B G G D D B A E E C F F C A

♤ 이진트리 운행 : 중위운행(좌,중,우) + 후위운행(좌,우,중)

```
// 방문되는 노드 순서에 대한 완성된 프로그램
#include<stdio.h>
typedef struct Node              // 이진트리 노드구조
{
  char data;                     // 노드에 저장된 문자
  struct Node *left, *right;     // 자식 포인터
} Node;

Node *newNode(int data)          // 새로운 노드 추가
{
    Node *node = (Node *)malloc(sizeof(Node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return(node);
}
```

```
void traverse(Node *node)          // 이진트리 운행
{
  if( !node ) return;              // 공트리이면 종료
  traverse(node->left);            // 좌측으로 이동
  printf(" %c", node->data);       // 중위운행으로 데이터 출력
  traverse(node->right);           // 우측으로 이동
  printf(" %c", node->data);       // 후위운행으로 데이터 출력
}

void main()
{
  Node *root = newNode('A');       // 루트노드 A 생성
  root->left = newNode('B');       // 좌측 자식노드 생성
  root->right = newNode('C');      // 우측 자식노드 생성
  root->left->right = newNode('D');
  root->right->left = newNode('E');
  root->right->right = newNode('F');
  root->left->right->left = newNode('G');
  printf("\n방문되는 노드 순서 : ");
  traverse(root);                  // 방문 순서 출력
}
```



// 문제 분석

- 주어진 문제는 **중위운행**과 **후위운행**이 **혼합**되어 있는 구조이다.
- **중위운행(좌,중,우)** : B G D A E C F
- **후위운행(좌,우,중)** : G D B E F C A

↓

↓ **중위**운행과 **후위**운행이 혼합되어 출력

↓

- **방문 순서** : B G G D D B A E E C F F C A

↓ ↓

↓ **후위운행**에서 루트 A는 **마지막**에 방문

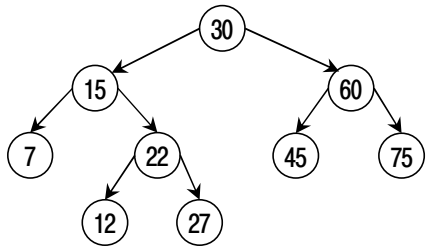
↓

중위운행에서 루트 A는 **중간**에 방문

정답 : ④

9. 〈보기 1〉의 이진트리(binary tree) T에 대한 설명으로 옳은 것을 〈보기 2〉에서 모두 고른 것은? [2022년 서울 7급]

----<보기 1>----



----<보기 2>----

ㄱ. 노드 22의 형제(sibling)는 노드 45이다.

ㄴ. T의 전위(preorder) 순회 결과는 노드 30 15 7 22 17 27 60 45 75 순이다.

ㄷ. T의 중위(inorder) 순회 결과는 노드 7 15 17 22 27 30 45 60 75 순이다.

ㄹ. T의 레벨 순서(level-order) 순회 결과는 노드 30 15 60 7 22 45 75 17 27 순이다.

---

① ㄱ, ㄴ, ㄷ

② ㄱ, ㄴ, ㄹ

③ ㄴ, ㄷ, ㄹ

④ ㄱ, ㄴ, ㄷ ,ㄹ

♤ 이진트리

• ㄱ. 노드 22의 형제(sibling)는 노드 45이다.(×)

→ 노드 22의 형제(sibling)는 노드 7이다.

정답 : ③