자료구조론	국가 전산 7급	2020년 9월 26일

♠ 필기합격인원/합격선(46명/76.66점) - 선발예정인원 33명 ♠

1. 다음의 D1~D4에서 제시된 키(key) 값의 순서대로 공백 이진탐색트리(binary search tree)에 데이터를 삽입하여 T1~T4를 만들었을 때, 모양이 다른 트리는? (단, 모양은 노드의 키 값과 간선의 연결구조를 포함한다) [2020년 국가 7급]

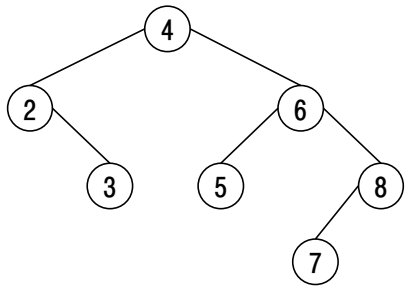
D1: 4, 2, 3, 6, 8, 5, 7 → 이진탐색트리 T1
D2: 4, 2, 6, 8, 5, 3, 7 → 이진탐색트리 T2
D3: 4, 6, 3, 8, 2, 5, 7 → 이진탐색트리 T3
D4: 4, 6, 8, 5, 7, 2, 3 → 이진탐색트리 T4

---

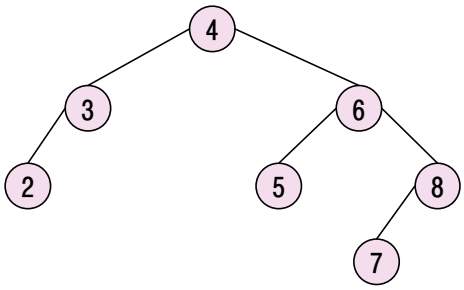
① T1 ② T2 ③ T3 ④ T4

♤ 이진탐색트리(binary search tree)

// 이런 유형의 문제는 주어진 데이터를 이용하여 직접 **이진탐색트리**를 그려 보아야 한다.



[T1, T2, T4의 이진탐색트리]



[T3의 이진탐색트리]

• 모양이 다른 이진탐색트리는 T3이다.

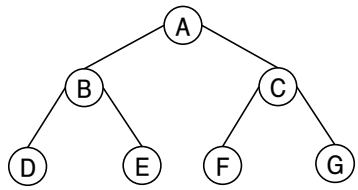
◈ 이진탐색트리 정의

이진탐색트리는 공백이 가능한 이진트리이며, 공백이 아니면 다음 조건을 만족한다.

• 모든 노드는 키를 가지며, 어떤 노드도 같은 키 값을 가지지 않는다.
• 왼쪽 서브트리의 키들(있으면)은 그 루트의 키보다 작은 값을 가진다.
• 오른쪽 서브트리의 키들(있으면)은 그 루트의 키보다 큰 값을 가진다.
• 왼쪽과 오른쪽의 서브트리는 모두 이진탐색트리이다.

정답 : ③

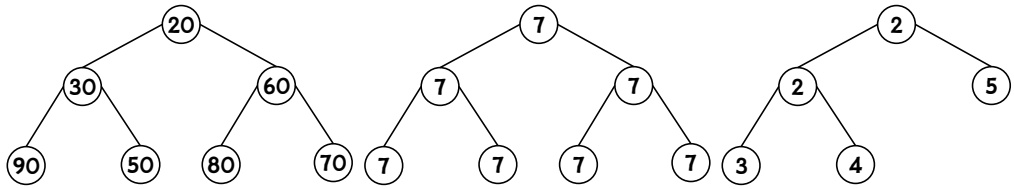
2. 다음 최소힙(min heap)에 대한 설명으로 옳지 않은 것은? [2020년 국가 7급]



① 완전이진트리(complete binary tree)이다.
② 노드 A는 가장 작은 키 값을 가져야 한다.
③ 노드 B와 노드 C가 같은 키 값을 가질 수 있다.
④ 노드 C는 노드 D보다 작은 키 값을 가져야 한다.

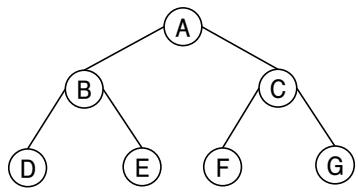
♤ **최소힙(min heap)**

// 다음 3개의 트리는 모두 **최소힙**이다.(부모노드 값≤자식노드 값)



[최소힙 1] **[최소힙 2]** **[최소힙 3]**

- **최소힙**은 완전이진트리이면서, 부모노드 값≤자식노드 값이다.
- **최소힙**은 부모노드 값이 가장 작다.



↓

- 노드 C는 노드 D보다 작은 키 값을 가져야 한다.(×)

↓

↓ 올바르게 고치면

↓

- 노드 C와 노드 D가 가지는 값은 **최소힙** 조건에 무관하다.

정답 : ④

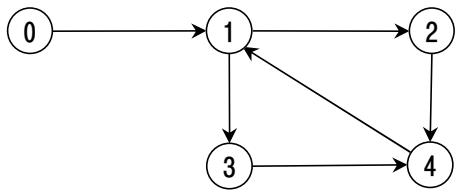
3. 다음 인접행렬(adjacency matrix)에 의해 표현되는 그래프에 대한 설명으로 옳은 것은? (단, 인접 행렬에서 [i]는 정점 i를 의미한다) [2020년 국가 7급]

	[0]	[1]	[2]	[3]	[4]
[0]	0	1	0	0	0
[1]	0	0	1	1	0
[2]	0	0	0	0	1
[3]	0	0	0	0	1
[4]	0	1	0	0	0

① 간선의 개수가 5개인 무방향그래프(undirected graph)이다.
② 정점 0에서 정점 4까지의 단순경로(simple path) 길이는 2이다.
③ 정점 3에 도달하기 위한 경로가 없는 정점이 있다.
④ 사이클(cycle)이 존재하는 방향그래프(directed graph)이다.

♤ 인접행렬(adjacency matrix)

// 그래프로 그리면 다음과 같다.



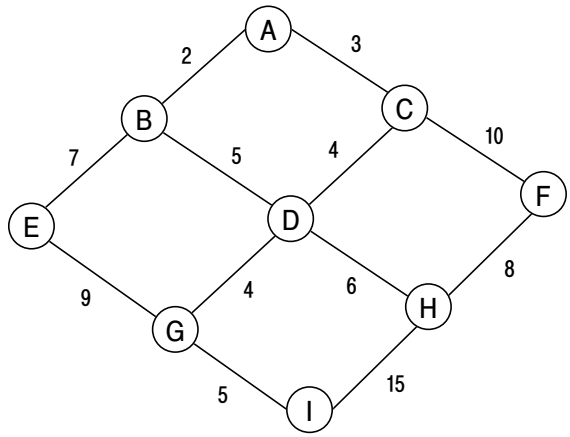
사이클(cycle)이 존재하는 방향그래프(directed graph)

↓
↓
↓

① 간선의 개수가 5개인 무방향그래프(undirected graph)이다.(×)
→ 간선의 개수가 6개인 방향그래프이다.
② 정점 0에서 정점 4까지의 단순경로(simple path) 길이는 2이다.(×)
→ 정점 0에서 정점 4까지의 단순경로(simple path) 길이는 3이다.
→ 정점 0에서 정점 4까지의 단순경로 : 0→1→2→4 또는 0→1→3→4
③ 정점 3에 도달하기 위한 경로가 없는 정점이 있다.(×)
→ 정점 3에 도달하기 위한 경로가 없는 정점은 없다.
→ (0, 1, 3) (1, 3) (2, 4, 1, 3) (4, 1, 3)

정답 : ④

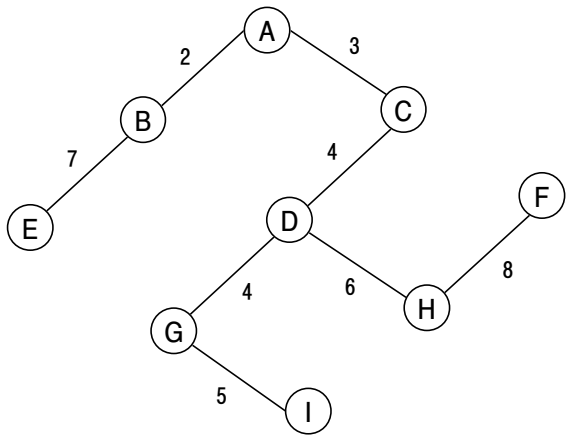
4. 다음 가중 그래프(weighted graph)에 Kruskal 알고리즘을 적용하여 구성한 최소비용 신장트리(minimum cost spanning tree)의 최소비용은? [2020년 국가 7급]



① 36 ② 39 ③ 4 ④ 46

♤ **최소비용 신장트리 – Kruskal 알고리즘**

- Kruskal 알고리즘은 그래프의 모든 간선 비용을 오름차순정렬하여 작은 간선부터 선택한다.
- 비용이 작은 간선부터 차례로 선택하면서, 사이클을 이루지 않으면 신장트리에 포함시킨다.
- 정점수가 n이면, 신장트리가 n-1 개의 간선을 가질 때까지 반복한다.



최소비용 신장트리

- 비용 5인 간선 (B, D)는 사이클 형성되므로 선택하지 않는다.
- **최소비용 = 2 + 3 + 4 + 4 + 5 + 6 + 7 + 8 = 39**

정답 : ②

5. 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?

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

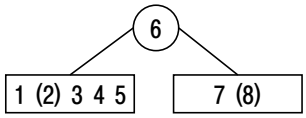
♤ **이진탐색트리 순회**

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

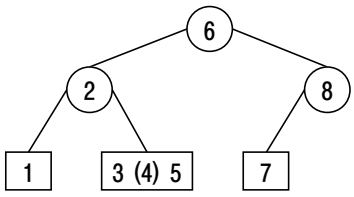
↓

• 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) : ?

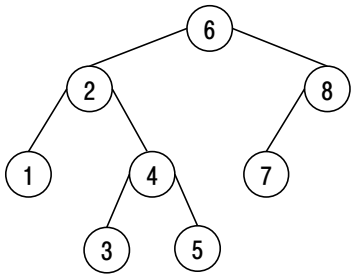
↓ 전위순회는 근노드를 가장 먼저 방문한다. **6**이 근노드라는 것을 알 수 있다.
↓ 중위순회는 근노드를 중간에 방문한다. 근노드를 중심으로 좌우측에 배치된다.



↓ 서브트리에서 2와 8이 근노드이므로



↓ 서브트리에서 4가 근노드이므로



완성된 이진탐색트리

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

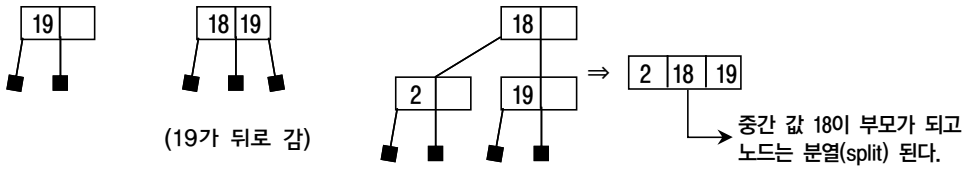
정답 : ③

6. 차수(m)가 3인 공백 B-트리에 19, 18, 2, 15, 6, 13, 7을 순서대로 삽입하여 구성한 B-트리에 대한 설명으로 옳은 것은? [2020년 국가 7급]

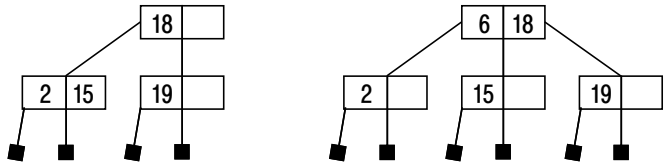
① 트리의 높이(height)가 1이다.
② 15는 루트(root) 노드에 저장되어 있다.
③ 3개의 자식(child) 노드를 가지는 노드가 존재한다.
④ 6이 저장된 노드와 18이 저장된 노드는 형제(sibling) 관계이다.

♤ **차수가 3인 공백 B-트리**

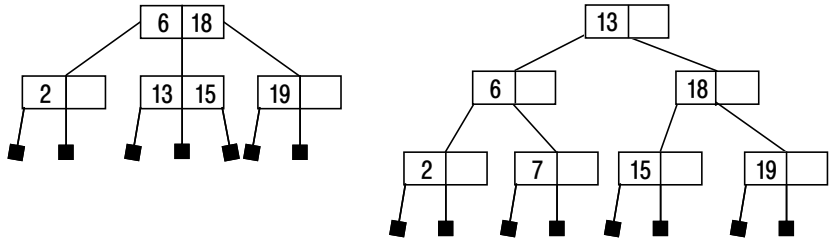
① 입력 19 ② 입력 18 ③ 입력 2 : 넘침 → 분열 → 새로운 노드 생성



④ 입력 15 ⑤ 입력 6 : 넘침 → 중간값 6이 부모로 이동 → 분열



⑥ 입력 13 ⑦ 입력 7 : 넘침 → 중간값 13이 부모로 이동 → 분열



• 6이 저장된 노드와 18이 저장된 노드는 **형제 관계**

정답 : ④

7. 스택을 이용하여 다음 후위표현식(postfix expression)의 연산을 수행하였다. 최종 결과 값과 연산 과정에서 스택 내 피연산자의 최대 개수를 바르게 연결한 것은? (단, *은 곱셈 연산자이고 /은 나눗셈 연산자이다) [2020년 국가 7급]

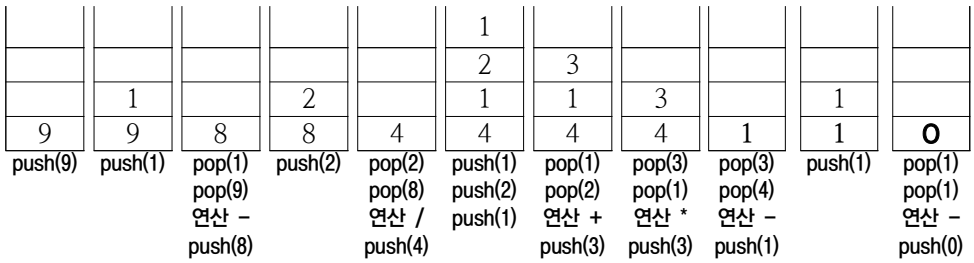
```
---------------------------------
 9 1 - 2 / 1 2 1 + * - 1 -
---------------------------------
```

	최종 결과 값	스택 내 피연산자의 최대 개수
①	0	3
②	0	4
③	1	3
④	1	4

♤ **후위표현식(postfix expression)**

- 후위표기식의 연산은 수식을 좌에서 우로 읽어가면서 연산자를 만났을 때 연산한다.
- 연산 대상은 연산자를 만나기 직전에 읽은 피연산자들이다.

// 후위표기식의 연산 과정을 Stack 그림으로 나타내면 다음과 같다.



- 최종 결과 값 = 0
- 스택 내 피연산자의 최대 개수 = 4

◈ **후위표기식의 연산 과정**

① 후위표기식의 수식을 왼쪽에서 오른쪽으로 차례로 읽으면서 하나씩 취한다.
- 읽은 자료가 피연산자이면 그냥 스택에 넣는다.(push)
- 읽은 자료가 연산자이면 스택에서 필요한 만큼의 피연산자를 꺼내(pop) 연산한다.
- 그리고, 연산된 결과만을 다시 스택에 넣는다.(연산은 나중에 출력된 피연산자가 앞에 온다)

② 위의 연산 과정 ①이 완료된 후, 스택에 저장된 최종 값이 수식의 연산 결과 값이다.

정답 : ②

8. 다음의 표는 단순연결리스트(singly linked list)를 표현한 것으로 각 행은 각 노드의 주소, 데이터 및 다음 노드 주소로 구성되어 있다. <다음 노드 주소 값>은 주소가 102인 노드를 삭제한 후 다음 노드 주소의 값을 설명한 것이다. ㉠~㉣에 들어갈 값을 바르게 연결한 것은? (단, 특정 노드의 다음 노드가 없을 때 그 노드의 다음 노드 주소 값은 NULL이다) [2020년 국가 7급]

주소	데이터	다음 노드 주소
100	LEE	NULL
101	PARK	104
102	KIM	101
103	JUNG	102
104	SEO	100

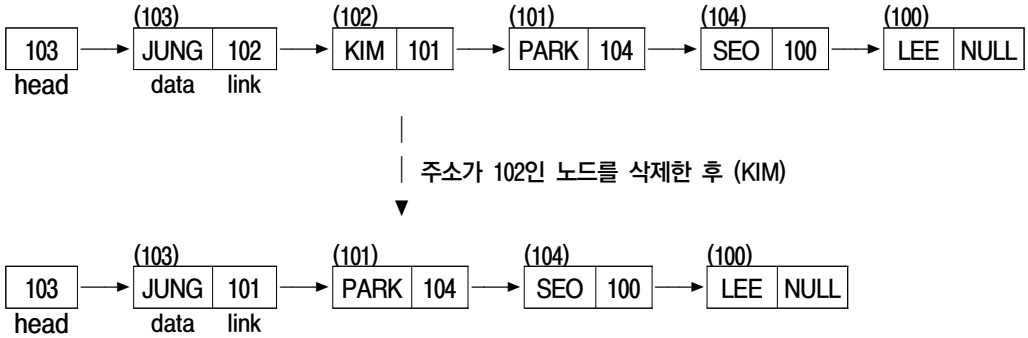
----<다음 노드 주소 값>----------

○ 주소가 100인 노드의 다음 노드 주소는 (㉠)이다.
○ 주소가 101인 노드의 다음 노드 주소는 (㉡)이다.
○ 주소가 103인 노드의 다음 노드 주소는 (㉢)이다.
○ 주소가 104인 노드의 다음 노드 주소는 (㉣)이다.

	㉠	㉡	㉢	㉣
①	NULL	104	101	100
②	NULL	103	101	104
③	101	103	102	104
④	101	104	102	100

♤ 단순연결리스트(singly linked list)에서 삭제

• 다음과 같은 단순연결리스트이다.(head의 주소는 **다음 노드 주소에 없는 값**이다)



정답 : ①

9. 다음 C 언어 프로그램의 출력 결과는? (단, 배열 h의 시작주소는 6487296이고, 자료형 int와 float는 4바이트의 크기를 가진다) [2020년 국가 7급]

```
int main() {
  struct human {
    char sextype;
    union t {
      char children[3];
      char beard[3];
    } u;
    int age;
    struct date {
      int month;
      int day;
      int year;
     } dob;
     int personalID;
     float salary;
  } h[10];
  printf("%d, %d, %d", sizeof(struct human),
                       &h[3].u.children[0] - &h[0].sextype,
                       &h[5].u.beard[1] - &h[0].sextype);
}
```

① 28, 85, 142 ② 28, 86, 143 ③ 31, 94, 157 ④ 31, 95, 158

♤ **구조체와 공용체**

• 먼저, 구조체 배열원소 하나의 메모리 크기 = 1 + 3 + 4 × 6 = **28**

	union t		struct date				
sextype	children[3] brard[3]	age	month	day	year	personalID	salary
1	3	4	4	4	4	4	4

→ 바이트 수

↳ 공용체 union t는 메모리를 공유(공용체 멤버 중 가장 큰 메모리만 할당)

// 구조체 배열 h[10]

h[0]	h[1]	h[2]	h[3]	h[4]	h[5]	h[6]	h[7]	h[8]	h[9]
6487296	6487324	6487352	6487380	6487408	6487436	6487464	6487492	6487520	6487548

→ 배열 각 원소 시작주소

• &h[3].u.children[0] − &h[0].sextype = (6487380 + 1) − (6487296) = **85**

• &h[5].u.beard[1] − &h[0].sextype) = (6487436 + 2) − (6487296) = **142**

정답 : ①

10. 다음 C 언어 프로그램의 출력 결과는? [2020년 국가 7급]

```
#include <stdio.h>
#define SWAP(x, y, t) (t)=(x), (x)=(y), (y)=(t)
void recursive(char *list, int i, int n)
{
  int j, temp;
  if (i == n)
  {
    for (j = 0; j <= n; j++)
      printf("%c", list[j]);
    printf(" ");
  }
  else
  {
    for (j = i; j <= n; j++)
    {
      SWAP(list[i], list[j], temp);
      recursive(list, i + 1, n);
      SWAP(list[i], list[j], temp);
    }
  }
}
int main()
{
  char list[3] = {'a', 'b', 'c'};
  recursive(list, 0, 2);
}
```

① ab ac bc
② ab bc ca
③ abc acb bac bca cba cab
④ abc acb bca bac cab cba

♤ 재귀호출을 이용한 순열(permutation)을 구하기

- 먼저, 주어진 프로그램은 문자열 abc에 대한 **순열(permutation)**을 구하는 문제이다.
- abc = {abc, acb, bac, bca, cba, cab}

// 문자열 abc에 대한 순열 알고리즘

① 인덱스 0번째 원소를 0번째부터 n-1번째까지 위치를 바꾼다.

• 문자열 abc에 대해 이 과정을 진행하면 3가지 종류 **abc, bac, cba**가 된다.

↓

• abc : 첫 번째[0] 원소 a를 첫 번째[0] 원소 a와 바꾼 것**(실제로는 바뀐 것이 없음)**

• bac : 첫 번째[0] 원소 a를 두 번째[1] 원소 b와 바꾼 것

• cba : 첫 번째[0] 원소 a를 세 번째[2] 원소 c와 바꾼 것

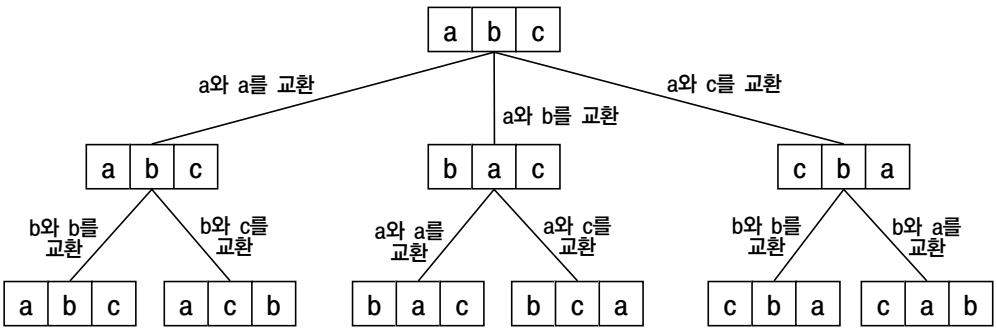
② 위의 ①번 과정의 결과에서 인덱스 1번째 원소를 1번째부터 n-1번째까지 위치를 바꾼다.

• abc : 두 번째[1] 원소 b를 두 번째[1] 원소 b와 바꾼 것(실제로는 바뀐 것이 없음) • acb : 두 번째[1] 원소 b를 세 번째[2] 원소 c와 바꾼 것
• bac : 두 번째[1] 원소 a를 두 번째[1] 원소 a와 바꾼 것(실제로는 바뀐 것이 없음) • bca : 두 번째[1] 원소 a를 세 번째[2] 원소 c와 바꾼 것
• cba : 두 번째[1] 원소 b를 두 번째[1] 원소 b와 바꾼 것(실제로는 바뀐 것이 없음) • cab : 두 번째[1] 원소 b를 세 번째[2] 원소 a와 바꾼 것

③ 이러한 과정을 n-1번 진행합니다.

↓

↓ 트리구조로 나타내면 다음과 같다.



↓ 원시코드(핵심부분)

```
for (int i = start; i <= end; i++)          //start=0, end=2인 경우
{
   swap(array[start], array[i]);            //시작부터 끝까지 모든 문자 교환
   permutation(array, start+1, end);        //시작 인덱스만 1 증가
   swap(array[start], array[i]);            //시작부터 끝까지 모든 문자 교환
}
```

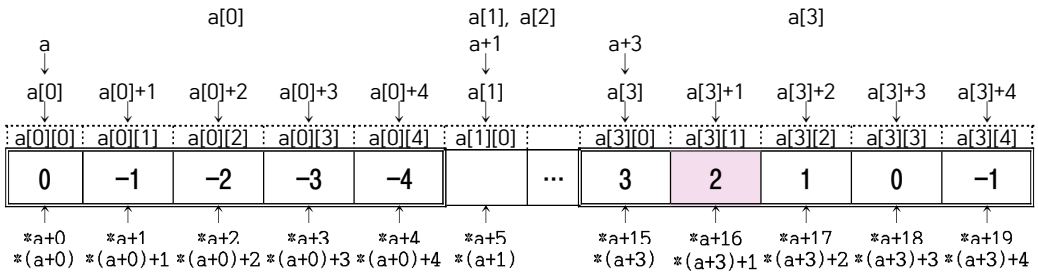
• 주어진 순열 알고리즘의 시간복잡도는 **O(n!)**이다.

정답 : ③

11. 행우선(row major) 순서로 저장되는 C 언어 2차원 배열 a[5][5]에서 다른 값을 갖는 것은? (단, 임의의 배열 인덱스(index) i(0≤i≤4)와 j(0≤j≤4)에 대해 a[i][j]가 (i−j)의 값을 가진다)

① a[3][1]
② *(*(a + 3) + 1)
③ *(a[3] + 1)
④ **((a + 3) + 1)

♤ **배열의 배열(2차원 배열)**



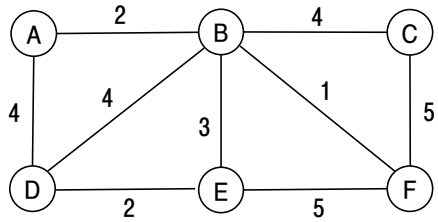
① a[3][1] = 2
② *(*(a + 3) + 1) = 2
③ *(a[3] + 1) = 2

④ **((a + 3) + 1) = **(a + 3 + 1) = **(a + 4) = 4
↓
안쪽 괄호는 없는 것과 같다.

// 배열의 배열에서 내용물을 나타내는 경우
- 대괄호가 2개 있는 경우 : a[3][1]
- 별이 2개 있는 경우 : *(*(a + 3) + 1)
- 괄호와 별이 각각 1개 있는 경우 : *(a[3] + 1)
  ↑
- 위의 내용만 알면, 시험장에서 답을 바로 찾을 수 있는 경우도 있다.
- 시험장에서 배열구조 그림을 그리지 않고, 답을 바로 찾을 수 있다는 것이다.

정답 : ④

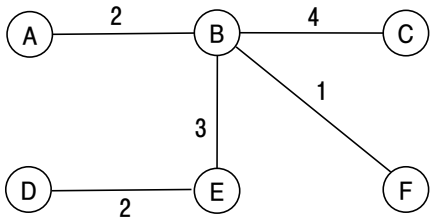
12. 다음 가중치 그래프에 Prim 알고리즘을 적용하여 최소비용 신장트리(minimum cost spanning tree)를 만들 때, 최소비용과 세 번째로 선택된 간선의 가중치의 합은? (단, A를 시작정점으로 한다) [2020년 국가 7급]



① 13 ② 14
③ 15 ④ 16

♤ **최소비용 신장트리 – Prim 알고리즘**

• 최소비용 신장트리는 다음과 같다.



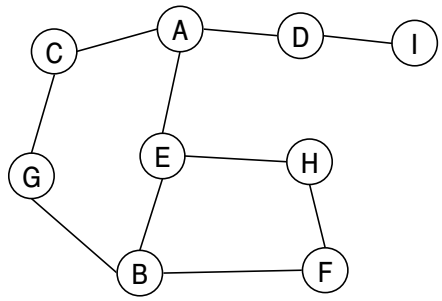
• 선택되는 간선 순서**(시작정점 A)** : (A, B) → (B, F) → **(B, E)** → (D, E) → (B, C)
• 최소비용 = 1 + 2 + 3 + 2 + 4 = **12**
• 세 번째로 선택된 간선 **(B, E)**의 가중치 = **3**
• 최소비용과 세 번째로 선택된 간선의 가중치의 합 = 12 + 3 = **15**

// Prim 알고리즘

① 그래프의 모든 간선 중에서 최소비용을 갖는 간선을 선택하여 트리에 포함시킨다.
② 선택된 간선이 (u, v)라 하면 정점 u와 v에 **연결된 간선 중 최소비용**을 갖는 간선을 선택한다.
③ 최소비용을 갖는 간선이 (v, w)라면 사이클이 형성되지 않으면 트리에 포함시킨다.
④ 다음은 정점 u, v, w에 연결된 간선 중에서 ②와 ③과 같은 기준으로 선택하여 트리에 포함시킨다.
⑤ 정점수가 n이면, 신장트리가 n-1 개의 간선을 가질 때까지 반복한다.

정답 : ③

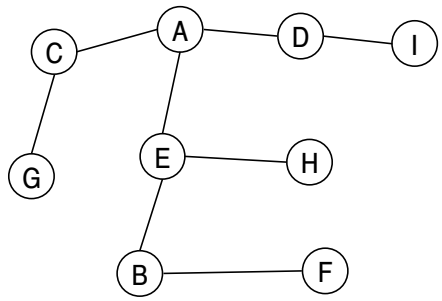
13. 다음 그래프에 대한 너비우선탐색(BFS: breadth first search)의 방문순서는? (단, 시작정점은 A이고, 인접한 정점들은 알파벳 순서로 방문한다) [2020년 국가 7급]



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

♤ 너비우선탐색(BFS: breadth first search)의 방문순서

• 너비우선탐색 결과는 다음과 같다.

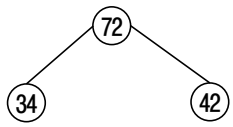


• BFS = A, C, D, E, G, I, B, H, F

// 너비우선탐색(BFS; breadth first search) → 큐를 이용
① 그래프에서 임의의 정점을 무작위로 선택한다. 만약, 정점 1이 선택되었다면
② 정점 1에 인접한 정점들 중 방문되지 않은 정점들을 임의의 순서로 방문한다.
③ 이때 정점 1에 인접한 정점들이 2, 3의 순서로 방문되었다면
④ 다음에는 먼저 방문한 정점 2에 인접하며 방문되지 않은 정점들을 방문하고
⑤ 이어서 정점 3의 순으로 인접한 정점들을 방문한다.

정답 : ①

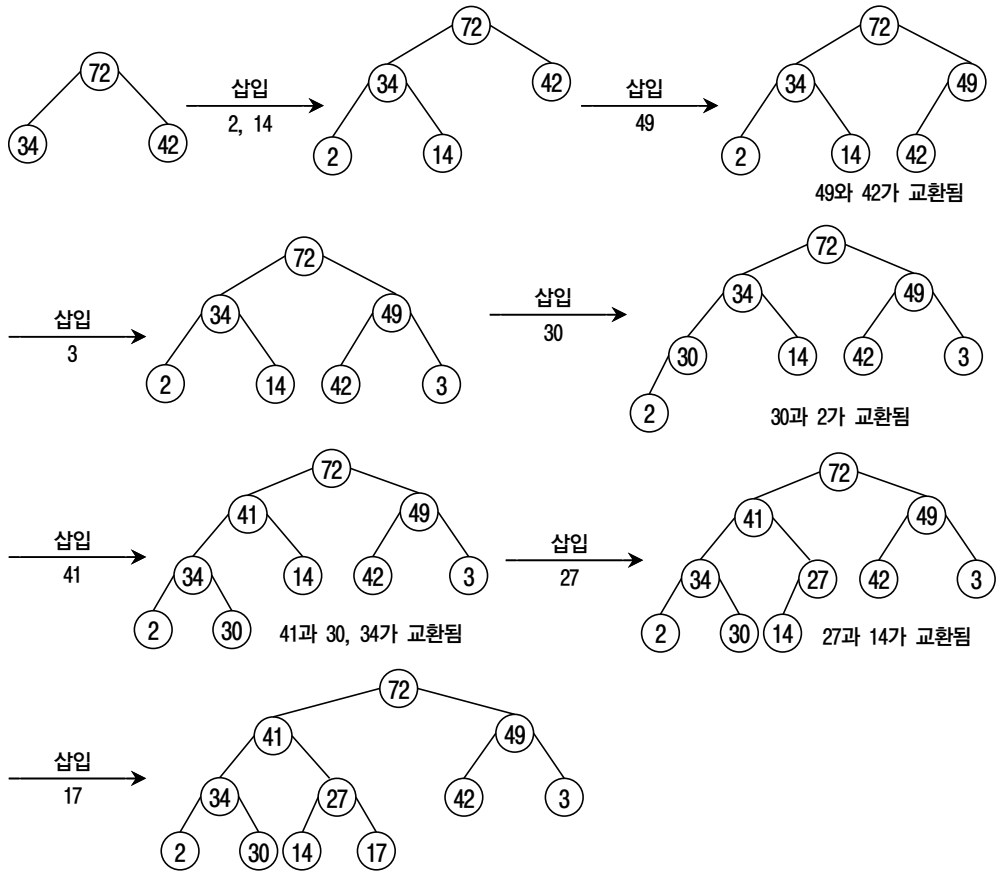
14. 다음에 주어진 최대힙(max heap)에 키 값이 2, 14, 49, 3, 30, 41, 27, 17인 8개의 데이터를 순서대로 삽입하여 최대힙을 구성하였다. 구성된 최대힙에서 모든 리프(leaf) 노드의 키 값의 합은? [2020년 국가 7급]



① 63 ② 73 ③ 108 ④ 13

♤ 최대힙(max heap)에 삽입

• 최대힙 : 부모노드 값≥자식노드 값 이면서, 완전이진트리이다.
• 추가 입력 : 2, 14, 49, 3, 30, 41, 27, 17



• 리프(leaf) 노드의 키 값 합 = 2 + 30 + 14 + 17 + 42 + 3 = 108

정답 : ③

15. 〈조건〉을 만족하고 front와 rear의 값이 모두 0인 원형큐(circular queue)에서 〈동작〉에 나열된 연산을 순서대로 모두 수행하였다. 연산 완료 후 원형큐에 존재하는 모든 유효 데이터와 front 및 rear의 값을 바르게 연결한 것은? (단, enqueue(x)에 의해 삽입된 후 dequeue()에 의해 삭제되지 않은 데이터만 유효 데이터로 인정한다) [2020년 국가 7급]

----<조건>----------

○ 원형큐는 원소 개수가 6개인 1차원 배열로 구현되었다.
○ front와 rear의 값이 같을 때 원형큐는 공백상태이다.
○ front와 (rear + 1) mod 6)의 값이 같을 때 원형큐는 포화상태이다.
○ enqueue(x)에 의해 rear의 값을 (rear + 1) mod 6)로 변경한 후 rear가 가리키는 위치에 데이터 x를 삽입한다. 단, 포화상태일 때에는 데이터 삽입이 이루어지지 않아 원형큐의 상태가 변하지 않는다.
○ dequeue()에 의해 front의 값을 (front + 1) mod 6)로 변경한 후 front가 가리키는 위치에서 데이터를 빼내어 삭제한다. 단, 공백상태일 때에는 데이터 삭제가 이루어지지 않아 원형큐의 상태가 변하지 않는다.

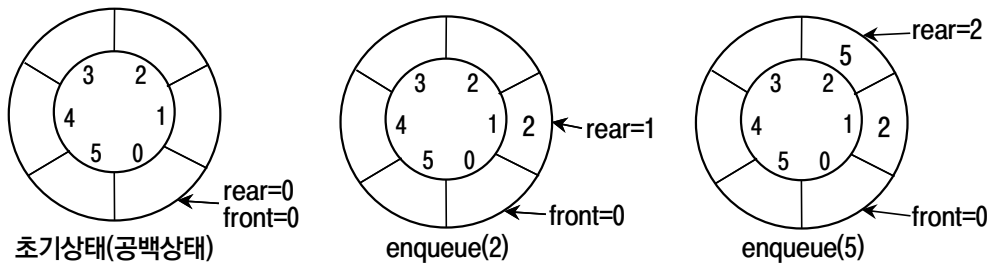
----<동작>----------

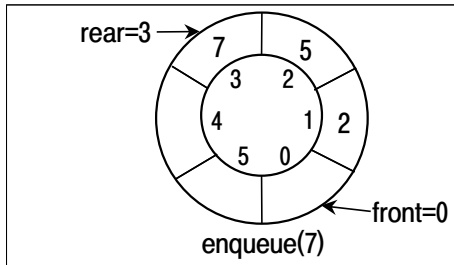
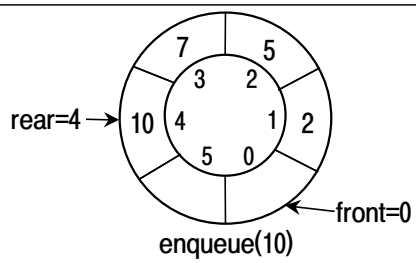
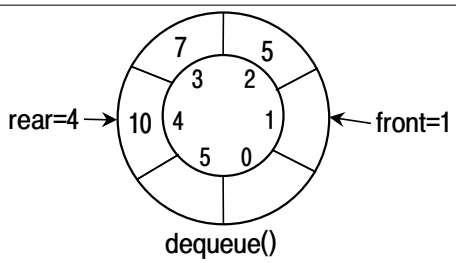
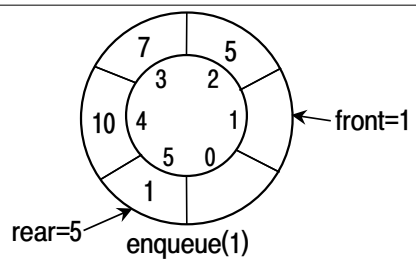
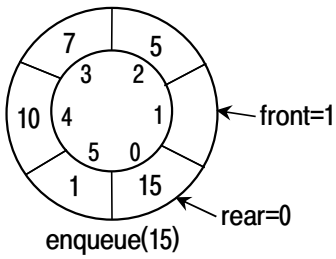
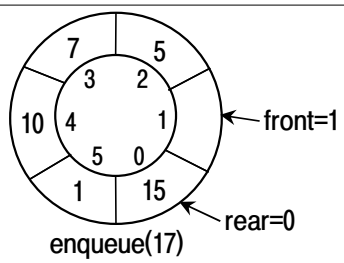
enqueue(2) → enqueue(5) → enqueue(7) → enqueue(10) → dequeue() → enqueue(1) → enqueue(15) → enqueue(17) → dequeue() → dequeue() → enqueue(20)

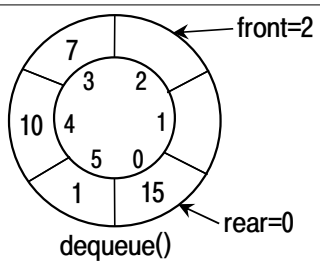
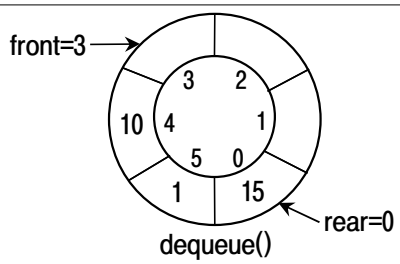
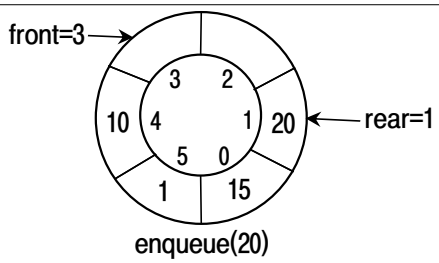
모든 유효 데이터	front	rear
① 10, 1, 15, 17, 20	3	1
② 10, 1, 15, 20	3	1
③ 10, 1, 15, 17, 20	4	2
④ 10, 1, 15, 20	4	2

♤ 공백조건의 원형큐(circular queue)

• 주어진 문제는 공백조건의 원형큐에 대한 삽입/삭제이다.
• 동작 과정은 다음과 같다.



<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td> 
</td>
<td> 

〈17은 포화상태이므로 입력 불가〉</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td> 
</td>
<td>• 모든 유효 데이터 = 10, 1, 15, 20
• front = 3
• rear = 1</td>
</tr>
</table>

• 주어진 문제는 문제 내용과 해설 풀이 과정만 길 뿐이다. 답을 쉽게 찾을 수 있었다.

정답 : ②

16. 키 값 1, 2, 3, 4, 5를 가지는 5개의 레코드로 이루어진 리스트를 퀵정렬(quick sort)로 오름차순정렬하고자 한다. 정렬 과정에서 총 비교횟수가 가장 많은 리스트는? (단, 피봇(pivot)은 정렬하고자 하는 대상의 첫 번째 레코드로 선택한다) [2020년 국가 7급]

① 1, 2, 3, 4, 5　　② 2, 3, 1, 4, 5
③ 3, 1, 2, 4, 5　　④ 3, 1, 2, 5, 4

♤ 퀵정렬(quick sort)

- 먼저, 퀵정렬의 평균 연산시간은 $O(n\log_2 n)$이다.
- 하지만, 퀵정렬은 최악의 경우 연산시간은 $O(n^2)$이다. - 정렬된 리스트
- 해서, 주어진 문제의 정답은 ①번이다. ①번은 자료가 정렬되어 있다.

// 퀵정렬에서 최악의 경우 수행시간 : $O(n^2)$ - 정렬된 리스트

초기 정렬자료 : [1 2 3 4 5]
1번째 분할 후 : 1 [2 3 4 5]
2번째 분할 후 : 1 2 [3 4 5]
3번째 분할 후 : 1 2 3 [4 5]
4번째 분할 후 : 1 2 3 4 [5]

- 퀵정렬에서 최악의 경우는 리스트가 분할되지 않을 때 발생된다. → 이미 정렬된 리스트

// ③ 3, 1, 2, 4, 5

	큰값(a++) → 1	2	3	← 작은값(b--) 4	5	left	right
초기배열	[3	1	2	4	5]	1	5
			↑ b	↑ a		**3과 2를 교환**	
1 단계	[2	1]	[3	4	5]		
		↑ a, b				**2와 1을 교환**	
2 단계	1	2	3	4	5		

- 참고로, 소량의 자료일 때에는 비교수가 명확하게 구분되는 것은 아니다.
- 아무튼, 주어진 문제는 정렬된 리스트에 대한 퀵정렬의 연산시간은 $O(n^2)$이다.
- 시험장에서는 이를 이용하여 답을 바로 찾을 수가 있다.

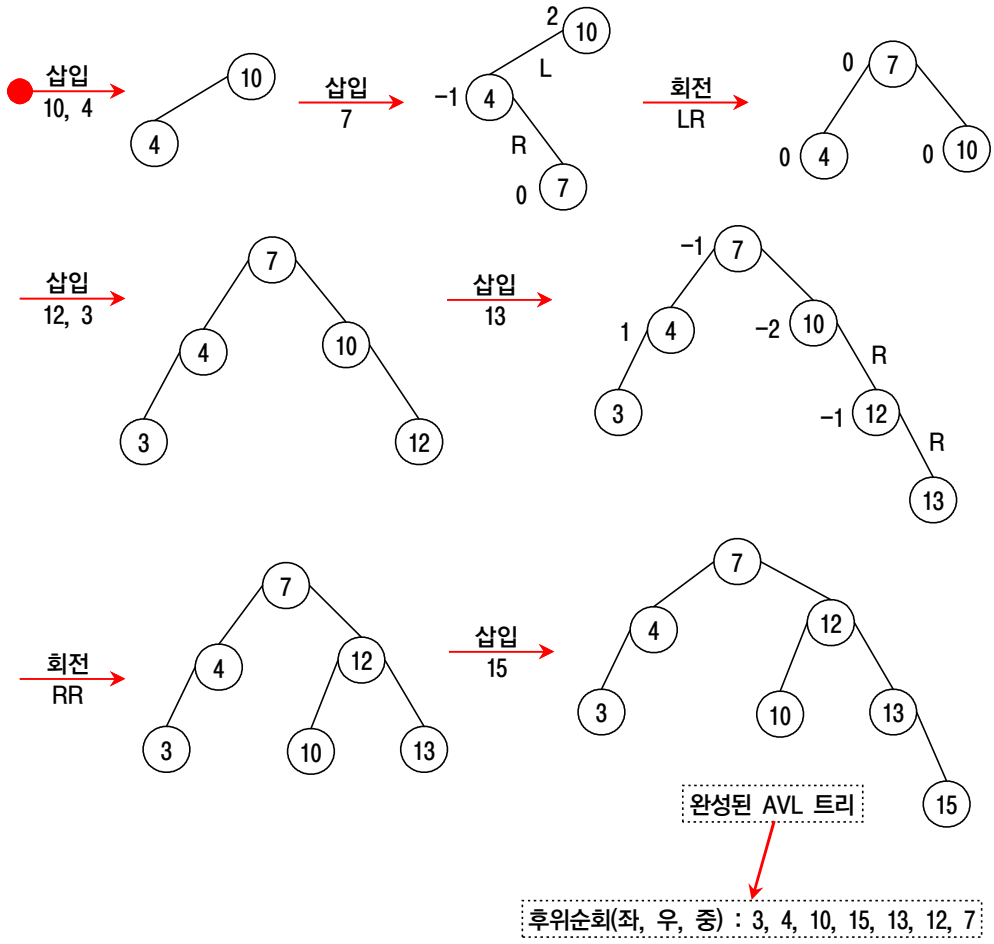
정답 : ①

17. 다음의 키 값을 가지는 7개의 데이터를 순서대로 삽입하여 AVL 트리를 구성한 후 후위순회를 수행하였다. 방문 노드들의 키 값을 방문 순서대로 바르게 나열한 것은? [2020년 국가 7급]

10, 4, 7, 12, 3, 13, 15
---

① 3, 4, 10, 15, 13, 12, 7
② 3, 7, 4, 12, 15, 13, 10
③ 3, 7, 4, 15, 13, 12, 10
④ 3, 10, 13, 15, 4, 12, 7

♤ AVL 트리



• 삽입 결과, 균형인수 값이 ±2인 노드가 2개 이상이 존재하면, 최근에 삽입된 노드로부터 가장 가까운 조상노드의 균형인수 값이 ±2인 노드를 기준으로 회전을 실시한다.

정답 : ①

18. 다음 C 언어 프로그램의 출력 결과는? [2020년 국가 7급]

```
#include <stdio.h>
void func(char s[], int size)
{
  if (size <= 0) return;
  s[size-1] = s[size-1] + 1;
  func(s, size-2);
}
int main()
{
  char str[] = "window";
  func(str, 6);
  printf("%c", str[3]);
}
```

① d ② e
③ n ④ o

♤ 재귀호출

// 프로그램 분석

```
void func(char s[], int size)    //size = 6
{
  if (size <= 0) return;         //완료조건
  s[size-1] = s[size-1] + 1;     //해당 인덱스 위치의 문자를 1 증가시킴
  func(s, size-2);               //인덱스 감소 : -2
}
```

↓메모리 구조

0	1	2	3	4	5	→ 인덱스(첨자)
w	i	n	d	o	w	
	↓ j		↓ e		↓ x	→ 인덱스 위치의 문자를 1 증가

• str[3] = e

정답 : ②

19. 다음은 1차원 배열 arr에서 두 번째로 큰 수를 찾는 C 언어 함수이다. ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은? (단, 배열 arr의 모든 원소는 서로 다른 양의 정수이고, n은 배열의 크기이며 2보다 크거나 같다) [2020년 국가 7급]

```
void Largest2nd(int *arr, int n){
  int i, max1 = 0, max2 = 0;
  for (i = 0; i < n; i+){
    if (    ㉠    ){
      max2 = max1;
      max1 = arr[i];
    }
    else if (    ㉡    )
      max2 = arr[i];
  }
  printf("Second Largest Number = %d\n", max2);
}
```

	㉠	㉡
①	arr[i] 〉 max1	arr[i] 〉 max2 && arr[i] 〈 max1
②	arr[i] 〉 max1	arr[i] 〈 max2 && arr[i] 〉 max1
③	arr[i] 〈 max1	arr[i] 〉 max2 && arr[i] 〈 max1
④	arr[i] 〈 max1	arr[i] 〈 max2 && arr[i] 〉 max1

♤ 1차원 배열 arr에서 두 번째로 큰 수를 찾는 C 언어 함수

```
void Largest2nd(int *arr, int n){                    //배열의 두 번째 큰 수를 찾는 함수
  int i, max1 = 0, max2 = 0;
  for (i = 0; i < n; i++){
    if ( arr[i] > max1 ){                            //배열의 값이 max1보다 크면
      max2 = max1;                                   //먼저, max2에 max1을 대입
      max1 = arr[i];                                 //배열의 값을 max1에 대입(가장 큰 값)
    }
    else if ( arr[i] > max2 && arr[i] < max1 )       //두 번째 큰 수를 찾기 위한 비교
      max2 = arr[i];                                 //배열의 두 번째 큰 수를 max2에 대입
  }
  printf("Second Largest Number = %d\n", max2);      //두 번째 큰 수 : 40
}
void main(){
  int a[] = {10, 20, 30, 40, 50};                    //배열 : 두 번째 큰 수는 40
  Largest2nd(a, 5);
}
```

정답 : ①

20. 다음은 원형연결리스트(circular linked list)의 맨 앞에 insert_node가 지정하는 노드를 삽입하는 C 언어 함수이다. ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은? (단, last_ptr은 마지막 노드를 지정하는 포인터 변수를 가리키는 주소 값을 가진다) [2020년 국가 7급]

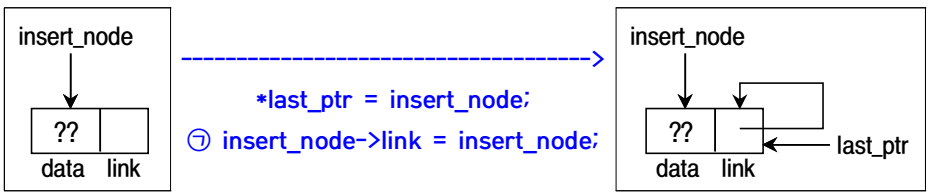
```
struct node
{
  int data;
  struct node *link;
};
typedef struct node *node_ptr;
void InsertFront(node_ptr *last_ptr, node_ptr insert_node)
{
  if (*last_ptr = NULL) {
    *last_ptr = insert_node;
          ㉠
  }
  else {
    insert_node->link = (*last_ptr)->link;
          ㉡
  }
}
```

	㉠	㉡
①	insert_node = insert_node->link;	(*last_ptr) = insert_node;
②	insert_node = insert_node->link;	(*last_ptr)->link = insert_node;
③	insert_node->link = insert_node;	(*last_ptr) = insert_node;
④	insert_node->link = insert_node;	(*last_ptr)->link = insert_node;

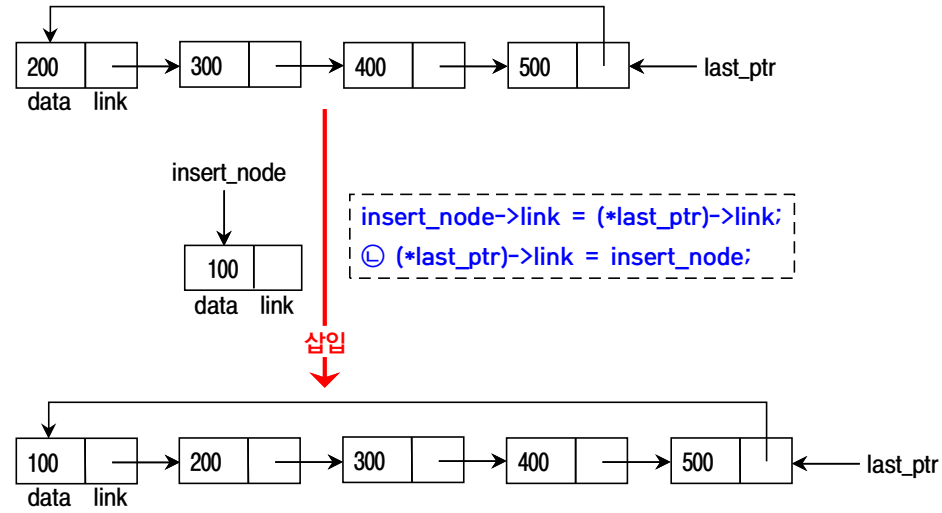
♤ 원형연결리스트의 맨 앞에 insert_node가 지정하는 노드를 삽입

// 공백인 경우

*last_ptr = NULL



// 공백이 아닌 경우


200
data
link
300
400
500
last_ptr
insert_node
100
data
link
insert_node->link = (*last_ptr)->link;
㉡ (*last_ptr)->link = insert_node;
삽입
100
data
link
200
300
400
500
last_ptr

정답 : ④