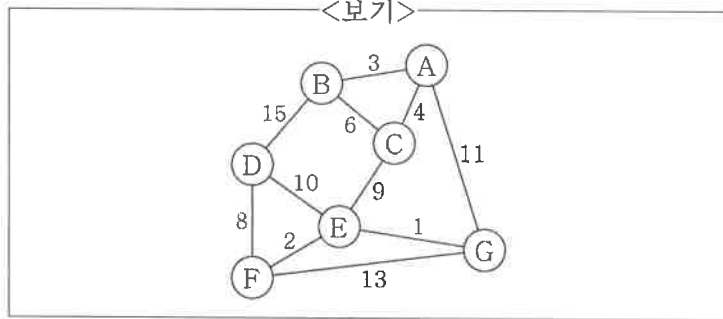


1. <보기>의 그래프에서 가중치가 낮은 간선부터 삽입해 나가는 크루스칼(Kruskal) 알고리즘을 이용하여 최소 비용 신장 트리를 구성하고자 할 때, 5번째로 삽입되는 간선의 가중치는?

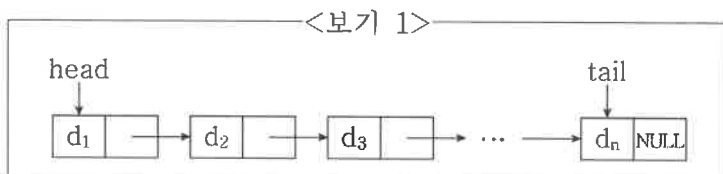


- ① 8
② 9
③ 10
④ 11

2. 배열로 표현한 이진 탐색 트리에 자연수를 저장할 때, 낭비되는 저장 공간이 가장 적은 입력 순서는?

- ① 6 - 7 - 12 - 19 - 3 - 22 - 17
② 12 - 19 - 17 - 6 - 3 - 7 - 22
③ 3 - 6 - 7 - 12 - 17 - 19 - 22
④ 22 - 19 - 17 - 12 - 7 - 6 - 3

3. <보기 1>과 같은 형태로 연결 리스트가 구성되어 있을 때, $O(1)$ 내에 수행할 수 없는 연산을 <보기 2>에서 모두 고른 것은? (단, head와 tail은 각각 첫 번째 원소와 마지막 원소를 가리키는 포인터이다.)



- <보기 2>
- ㄱ. 첫 번째 위치에 새로운 원소 삽입
ㄴ. 마지막 위치에 새로운 원소 삽입
ㄷ. 마지막 위치의 원소 삭제
ㄹ. d_1 부터 d_n 중에 특정 값이 저장되어 있는지 확인

- ① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄷ, ㄹ

4. <보기>의 배열 A를 버블 정렬과 삽입 정렬을 이용하여 오름차순으로 정렬하려고 한다. (가) 버블 정렬과 (나) 삽입 정렬에 의한 원소 교환 횟수를 가장 옳게 짝지은 것은? (단, 삽입 정렬에서 배열 원소의 이동은 인접한 원소 교환에 의해서만 수행된다. 예를 들어, A[5]를 A[2]와 A[3] 사이에 삽입하기 위해서는 두 번의 교환 연산 $A[5] \leftrightarrow A[4]$, $A[4] \leftrightarrow A[3]$ 을 수행한다.)

<보기>

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
A	5	8	3	1	4	6	7	2

- | | | |
|---|-----|-----|
| | (가) | (나) |
| ① | 7번 | 7번 |
| ② | 7번 | 15번 |
| ③ | 15번 | 7번 |
| ④ | 15번 | 15번 |

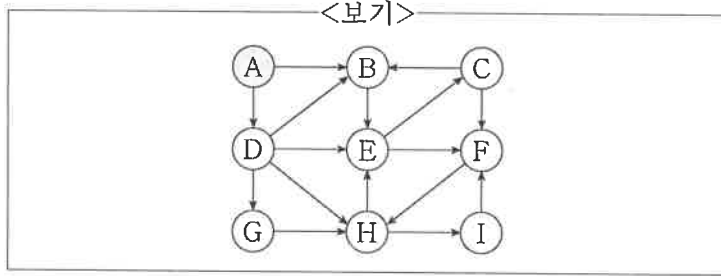
5. 재귀를 이용한 이진 탐색(binary search)으로 3을 찾을 때, 함수 호출 횟수가 가장 많은 배열은? (만약, 이진 탐색할 범위의 길이가 $2n+1$ 이라면 $n+1$ 번째 원소가, $2n$ 이라면 n 번째 원소가 기준이 된다.)

- ① 1 2 3 4 5 6
② 2 3 4 5 6 7
③ 3 4 5 6 7 8
④ -1 0 1 2 3 4

6. 제시된 인접 행렬(adjacency matrix) 중, 그래프로 구현하였을 때 사이클(cycle)이 생성되는 것은?

- | | |
|--|--|
| ① $\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ | ② $\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$ |
| ③ $\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ | ④ $\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ |

7. <보기>의 그래프에서 정점 A를 출발점으로 너비 우선 탐색(breadth first search)을 수행하는 과정에서 나타날 수 있는 큐(queue)의 상태로 가장 옳지 않은 것은?



- ① (앞) D - E (뒤)
 ② (앞) C - I - F (뒤)
 ③ (앞) B - E - G - H (뒤)
 ④ (앞) H - G - C - F (뒤)

8. <보기>에서 수식 트리와 수식 표기법에 대한 설명으로 옳은 것을 모두 고른 것은?

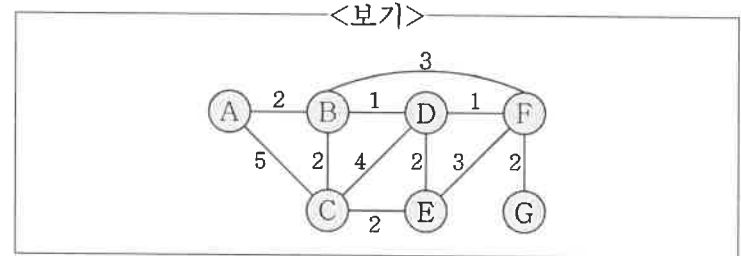
- <보기>
- ㄱ. 중위 표기식을 후위 표기식으로 변환할 때, 피연산자를 저장하기 위한 스택을 보조 자료구조로 사용한다.
 ㄴ. 중위 표기식과 이에 대한 후위 표기식에서 피연산자가 나타나는 순서는 동일하다.
 ㄷ. 수식 트리를 후위 순회하면 후위 표기식을 얻을 수 있다.
 ㄹ. 후위 표기식을 계산할 때, 수식의 가장 끝에 있는 연산자가 가장 마지막에 계산되지 않을 수도 있다.

- ① ㄱ, ㄴ
 ② ㄱ, ㄷ
 ③ ㄴ, ㄷ
 ④ ㄷ, ㄹ

9. 크기가 13인 원형 큐(circular queue)를 배열로 구현할 때, 큐에 원소가 가장 많이 들어있다고 판단되는 경우는? (단, front와 rear는 배열의 인덱스 번호이며, front는 바로 다음에 삭제될 원소의 인덱스, rear는 새로 들어올 원소가 저장될 인덱스이다.)

- | | front | rear |
|---|-------|------|
| ① | 5 | 3 |
| ② | 3 | 5 |
| ③ | 5 | 1 |
| ④ | 1 | 5 |

10. 다익스트라(Dijkstra) 알고리즘을 이용하여 <보기> 그래프의 정점 A에서 각 정점으로의 최단 경로를 구할 때, 최단 경로를 발견하는 순서를 바르게 나열한 것은? (만약, 둘 이상의 정점에 대해 동시에 최단 경로를 찾을 수 있다면, 알파벳 순으로 정렬한다.)



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

11. <보기>의 항목들의 시간 복잡도를 낮은 것부터 높은 것 순서대로 바르게 나열한 것은?

- <보기>
- ㄱ. 임의의 자연수 n과 정수 a, b, c를 입력으로 받는 다음 함수

```
void hanoi(int n, int a, int b, int c) {
    if (n >= 1) {
        hanoi(n-1, a, c, b);
        printf("%d to %d\n", a, b);
        hanoi(n-1, c, b, a);
    }
}
```

- ㄴ. 시간 복잡도 함수 $T(n) = 5n^4 + 2n^3 + n$
 ㄷ. 배열로 표현한 스택(stack)에서의 삭제(pop) 연산 수행
 ㄹ. 단순 연결 리스트에서 임의의 원소를 탐색하는 연산 수행 중 최악의 경우

- ① ㄱ - ㄴ - ㄹ - ㄷ
 ② ㄷ - ㄱ - ㄹ - ㄴ
 ③ ㄷ - ㄹ - ㄴ - ㄱ
 ④ ㄹ - ㄷ - ㄱ - ㄴ

12. 배열로 구현된 이중 연결 리스트(doubly linked list)에서 배열 A는 원소의 키값인 정수를 저장하고 있고, 배열 prev, next는 각각 이중 연결 리스트에서 직전 원소와 다음 원소가 저장된 배열의 인덱스를 저장하고 있다. prev와 next에 0이 저장되어 있다면 이는 NULL값을 의미한다. 리스트를 순서대로 방문했을 때, 저장된 키값이 점점 커지는 것은?

①	인덱스	1	2	3	4	5	6
	A	1	3	2	5	4	6
	prev	0	3	1	5	2	4
	next	3	5	2	6	4	0

②	인덱스	1	2	3	4	5	6
	A	1	2	3	4	5	6
	prev	2	3	4	5	6	0
	next	0	1	2	3	4	5

③	인덱스	1	2	3	4	5	6
	A	1	3	2	5	4	6
	prev	3	5	2	6	4	0
	next	0	3	1	5	2	4

④	인덱스	1	2	3	4	5	6
	A	1	2	3	4	5	6
	prev	2	4	5	3	6	0
	next	0	1	4	2	3	5

13. <보기>는 어떤 이진 탐색 트리의 키값을 후위 순회 방문 순서대로 나열한 것이다. 이 트리에 대한 설명으로 가장 옳은 것은?

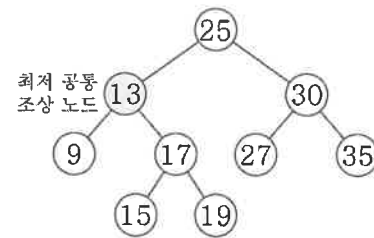
<보기>

21, 39, 50, 47, 32, 55, 52, 80, 91, 85, 79, 67

- ① 39는 내부 노드(비단말 노드)이다.
 ② 47의 왼쪽 자식은 32이다.
 ③ 79는 자식을 하나만 가진다.
 ④ 루트 노드의 왼쪽 자식은 55이다.

14. 이진 탐색 트리에 속한 두 개의 노드(node)가 주어질 때, 이들의 최저 공통 조상(lowest common ancestor) 노드를 찾는 코드를 작성하려고 한다. 이때, 최저 공통 조상 노드란 임의의 두 노드가 공통으로 갖는 조상 중, 가장 가까운 노드를 의미한다. 예를 들어, <보기 1>의 트리에서 노드 9와 19의 최저 공통 조상 노드는 13이다. <보기 2>의 (가)와 (나)에 작성해야 할 코드를 옳게 짠 것은? (단, 최저 공통 조상 노드를 찾는 두 개의 노드는 이미 트리에 존재한다고 가정한다.)

<보기 1>



<보기 2>

```
typedef struct Node {
    struct Node* left;
    struct Node* right;
    int value;
} Node;

Node* findLCA(Node* root, int val1, int val2) {
    while (root != NULL) {
        int value = root->value;
        if ( (가) ) {
            root = root->left;
        } else if ( (나) ) {
            root = root->right;
        } else {
            return root;
        }
    }
    return NULL; // 트리가 비어있을 경우만
}

/* child1: 주어진 첫 번째 노드 (예시에서는 노드 9),
   child2: 주어진 두 번째 노드 (예시에서는 노드 19) */
Node* findLCAwithNodes(Node* root, Node* child1, Node* child2) {
    if (root == NULL || child1 == NULL || child2 == NULL) {
        return NULL;
    }

    return findLCA(root, child1->value, child2->value);
}
```

- ① (가) value < val1 && value < val2
 (나) value > val1 && value > val2
 ② (가) value > val2 && value < val1
 (나) value < val2 && value < val1
 ③ (가) value > val1 && value > val2
 (나) value < val1 && value < val2
 ④ (가) value > val1 && value < val2
 (나) value < val1 && value < val2

15. <보기>는 힙 정렬(heap sort) 알고리즘을 최대 힙(max heap)로 구현한 C언어 코드이다. (가), (나)에 들어갈 코드를 옳게 짤지은 것은? (단, 함수 swap()은 주어진 두 인자의 값을 서로 교환하는 사용자 정의 함수라고 가정한다.)

<보기>

```
void adjust(int a[], int root, int n){
    int child, rootkey;
    int temp;
    temp = a[root];
    rootkey = a[root];
    child = 2 * root;
    while (child <= n){
        if ((child < n) &&
            (a[child] < (a[child+1]))
            child++;
        if ( (가) )
            break;
        else {
            (나);
            child *= 2;
        }
    }
    a[child/2] = temp;
}

void heapSort(int a[], int n){
    int i;

    for (i = n/2; i > 0; i--)
        adjust(a, i, n);
    for (i = n; i > 1; i--){
        swap(a[1], a[i]);
        adjust(a, 1, i-1);
    }
}
```

- | | |
|----------------------|-----------------------|
| (가) | (나) |
| ① rootkey < a[child] | a[child] = a[child/2] |
| ② rootkey < a[child] | a[child/2] = a[child] |
| ③ rootkey > a[child] | a[child] = a[child/2] |
| ④ rootkey > a[child] | a[child/2] = a[child] |

16. <보기 1>의 배열 A는 최소 힙(min heap)을 나타낸다. 이 힙에 <보기 2>의 연산을 차례로 수행했을 때, 배열 A의 최종 결과로 가장 옳은 것은?

<보기 1>

0	1	2	3	4	5	6	7	8	
A		5	30	10	40	60	70	20	50

(0번 원소는 사용하지 않음)

<보기 2>

1. 최소값 삭제
2. 5 삽입

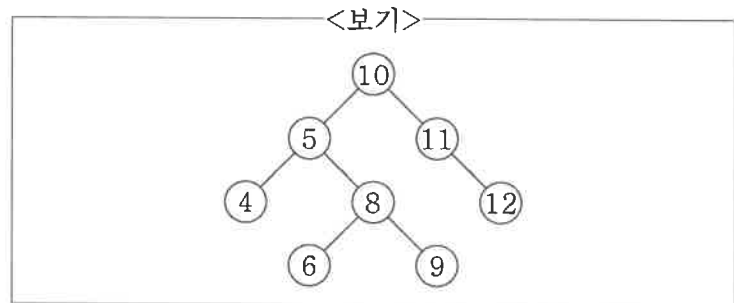
- | | | | | | | | | | |
|-----|---|---|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| ① A | | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
- | | | | | | | | | | |
|-----|---|---|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| ② A | | 5 | 10 | 20 | 30 | 60 | 70 | 50 | 40 |
- | | | | | | | | | | |
|-----|---|---|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| ③ A | | 5 | 30 | 10 | 40 | 60 | 70 | 20 | 50 |
- | | | | | | | | | | |
|-----|---|---|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| ④ A | | 5 | 30 | 10 | 40 | 50 | 20 | 70 | 60 |

17. 크기가 11인 배열 기반의 해시 테이블에 <보기>의 해시 함수 h1(k)와 h2(k)를 사용하는 이중 해싱(double hashing)으로 입력 키를 차례대로 저장할 때 39가 저장되는 테이블의 주소는?

- <보기>
- 기본 해시 함수 $h1(k) = k \bmod 11$
 - 2차 해시 함수 $h2(k) = 1 + (k \bmod 10)$
 - 입력 키: 12, 44, 13, 88, 23, 94, 11, 39

- ① 2
- ② 3
- ③ 4
- ④ 9

18. <보기>의 AVL 트리에 7을 삽입하고, 높이 균형을 유지하기 위해 회전 연산을 수행하였다. 이때, 변환된 AVL 트리에 대한 층위 순회(level-order) 방문 순서는?



- ① 5, 4, 6, 7, 8, 9, 10, 11, 12
- ② 5, 4, 6, 7, 9, 8, 12, 11, 10
- ③ 10, 6, 11, 4, 8, 7, 9, 5, 12
- ④ 10, 6, 11, 5, 8, 12, 4, 7, 9

19. <보기 1>은 인자로 들어온 두 집합의 교집합을 구하여 반환하는 재귀함수이다. (가)와 (나)에 들어갈 내용을 가장 옳게 짝지은 것은? (단, 인자로 들어오는 두 집합과 결과인 교집합 모두 <보기 2>의 예시와 같이 연결 리스트로 표현된다.)

<보기 1>

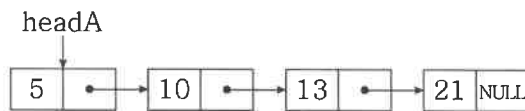
```
typedef struct Node {
    int data;
    struct Node *next;
} Node;    // 노드 정의

// headA와 headB: 두 집합을 표현하는 연결 리스트의 첫 노드의 주소
// 반환 값 p: 두 집합의 교집합을 표현하는 연결 리스트의 첫 노드의 주소
Node *intersect(Node *headA, Node *headB) {
    if (headA == NULL || headB == NULL) // 베이스 케이스
        return NULL;

    if (headA->data > headB->data)
        (가)
    else if (headA->data < headB->data)
        (나)
    else {
        Node *p = (Node *) malloc(sizeof(Node));
        p->data = headA->data;
        p->next = intersect(headA->next, headB->next);
        return p;
    }
}
```

<보기 2>

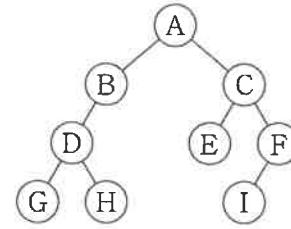
집합 A={5, 10, 13, 21}을 표현하는 연결 리스트
(원소는 오름차순으로 저장)



- ① (가) intersect(headA, headB->next);
 (나) intersect(headA->next, headB);
 ② (가) intersect(headA->next, headB);
 (나) intersect(headA, headB->next);
 ③ (가) return intersect(headA, headB->next);
 (나) return intersect(headA->next, headB);
 ④ (가) return intersect(headA->next, headB);
 (나) return intersect(headA, headB->next);

20. <보기 1>의 이진 트리의 루트(root)가 A노드일 때, <보기 2>의 traverse_X(root)를 수행하였다. 노드의 출력 순서로 가장 옳은 것은?

<보기 1>



<보기 2>

```

alg traverse_X(노드 v) {
    if (v == NULL) return;
    노드 v 출력;
    traverse_Y(v의 왼쪽 자식);
    traverse_Y(v의 오른쪽 자식);
}

alg traverse_Y(노드 v) {
    if (v == NULL) return;
    traverse_X(v의 왼쪽 자식);
    traverse_X(v의 오른쪽 자식);
    노드 v 출력;
}

```

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