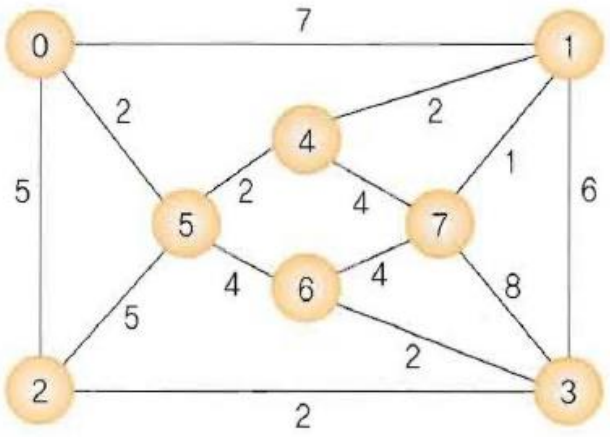
Sejong Univ., Algorithms, Mid-term Test, Oct. 23, 2014, D.N. Kim

Name ( ) Student ID Number ( ) **만점 75**

1. (5점) 아래의 그래프에 0번 정점을 시작 정점으로 Dijkstra의 알고리즘을 이용해 최단 경로를 구하는 과정을 배열 distance[ ]의 변화 과정으로 보여라.



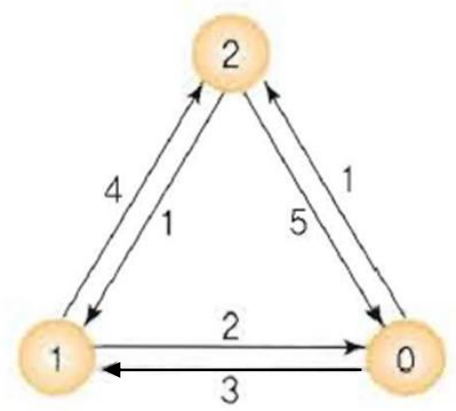
단계	선택된 정점	found[ ] 배열								distance[ ] 배열							
		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
1	**0**	1								0	7	5	INF	INF	2	INF	INF
2	5	1					1			0	7	5	INF	4	**2**	6	INF
3	4	1				1	1			0	6	5	INF	**4**	2	6	8
4	2	1		1		1	1			0	6	**5**	7	4	2	6	8
5	1	1	1	1		1	1			0	**6**	5	7	4	2	6	7
6	6	1	1	1		1	1	1		0	6	5	7	4	2	**6**	7
7	3	1	1	1	1	1	1	1		0	6	5	**7**	4	2	6	7
8	7	1	1	1	1	1	1	1	1	0	6	5	7	4	2	6	**7**

무한대 값을 대신해서 INF 이용한다.

distance[ ] 배열 에 저장된 가중치가 동일 한 경우 정점 순서는 1,2,3,4 순으로 선택 (for 문에서 그렇게 되어있으므로)

※ 정점 7 맞으면 0.5점 추가 , 반만 맞으면 2.5

2. (5점) 아래 그래프에 Floyd의 알고리즘을 이용해 최단 경로를 구할 때 2차원 배열 A[ ][ ] 가 변경되는 모습을 보여라.



	0	1	2
0	0	3	1
1	2	0	4
2	5	1	0

$A^{-1}[i][j]$

	0	1	2
0	0	3	1
1	2	0	3
2	5	1	0

$A^{0}[i][j]$

	0	1	2
0	0	3	1
1	2	0	3
2	3	1	0

$A^{1}[i][j]$

	0	1	2
0	0	2	1
1	2	0	3
2	3	1	0

$A^{2}[i][j]$

인접행렬에 해당됨

미비하면 2.5

하나 틀리면 1.2x3 = 3.8

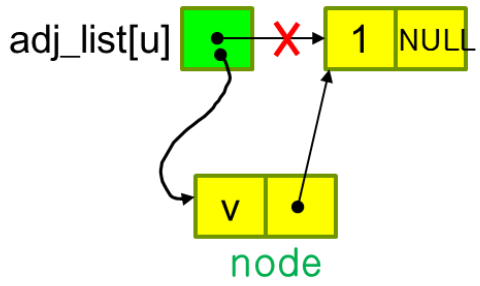
3. (5점) 인접 리스트를 이용한 그래프 추상 데이터 타입의 구현이다.

```
// 간선 삽입 연산, v를 u의 인접 리스트에 삽입한다.
void insert_edge(GraphType *g, int u, int v)
{
        GraphNode *node;
        if( u >= g->n || v >= g->n ){
                fprintf(stderr, "그래프: 정점 번호 오류");
                return;
        }
        node = (GraphNode *)malloc(sizeof(GraphNode));
        node->vertex = v;  // 아래 줄과 교환 가능함
        node->link = g->adj_list[u];
        g->adj_list[u] = node;
}
```

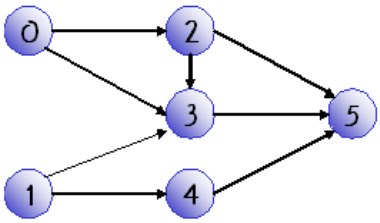
(아래) 에서 골라서 줄 별 바른 순서로 쓰시오. 밑에 있는 그림을 참고하시오.

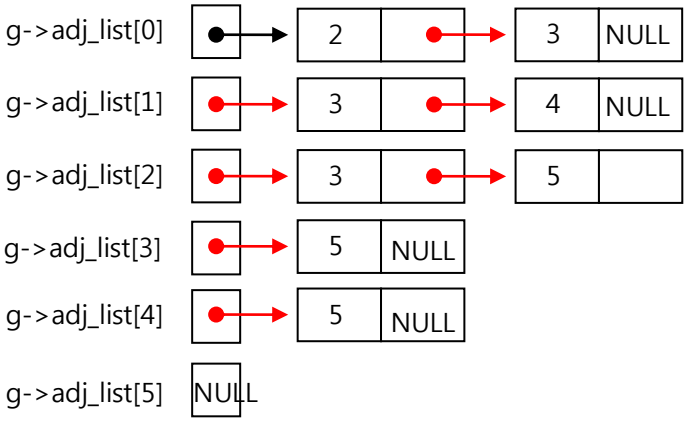
```
node->link = g->adj_list[u];
node = (GraphNode *)malloc(sizeof(GraphNode));
g->adj_list[u] = node;
node->vertex = v;
```

답: 2-1-3 맞음, 2-3-1 틀림 틀림, 4는 순서 상관 없음



4. 위상 정렬에 대한 문제이다.





(1) (5점) 그래프를 인접 리스트로 표현하라. 즉 위 그림을 완성하라.
연결 리스트의 각 노드는 구조체 타입 GraphNode로 정의되어있으니 그릴 때 참고바람.

```
typedef struct GraphNode
{
        int vertex;
        struct GraphNode *link;
} GraphNode;
typedef struct GraphType
{
        int n;
        GraphNode *adj_lst[MAX_VERTICES];
} GraphType;
```

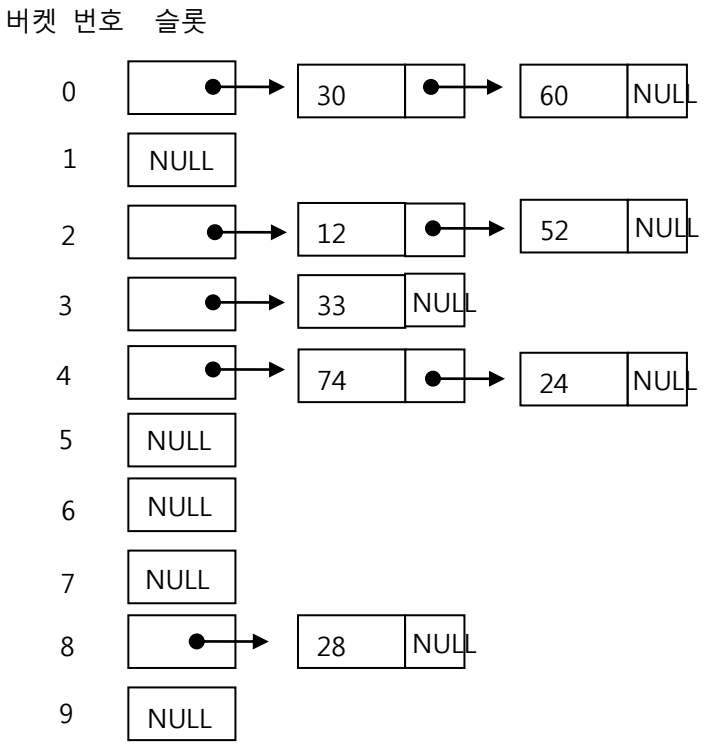
```
void topo_sort(GraphType *g)
{
    int i;
    StackType s;
    GraphNode *node;
    // 모든 정점의 진입 차수를 계산
    int *in_degree = (int *)malloc(g->n* sizeof(int));
    for(i = 0; i < g->n; i++)                    // 초기화
        in_degree[i] = 0;
    for(i = 0; i < g->n; i++) {
        GraphNode *node = g->adj_list[i];
        while ( node != NULL ) {
            in_degree[node->vertex]++;
            node = node->link;
        }
    }
```

(2) (5점) 위 프로그램에서 in_degree[ ] 는 입력 차수를 저장하는 배열이다. for 문의 각 단계에서 in_degree[ ] 에 저장된 값을 쓰시오.

	0	1	2	3	4	5
i=0			1	1		
i=1			1	2	1	
i=2			1	3	1	1
i=3			1	3	1	2
i=4			1	3	1	3
i=5	0	0	1	3	1	3

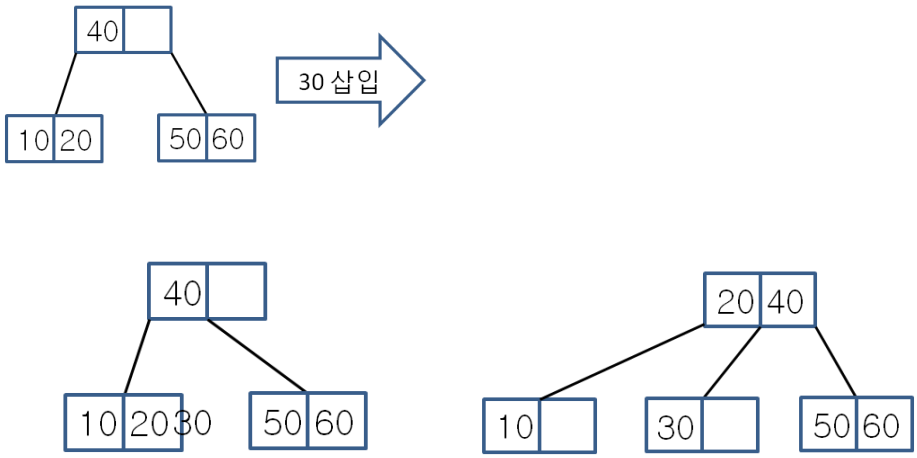
5. (5점) 아래와 같이 10개의 버켓을 가지는 해시 테이블이 있다. 해시 함수는 정수 데이터의 끝자리 숫자를 사용하고, 체이닝으로 오버플로우를 처리한다. 데이터가 순서대로 삽입될 경우에 해시 테이블에 데이터가 저장되는 과정을 보여라.

   데이터: 74, 30, 12, 28, 24, 33, 52, 60

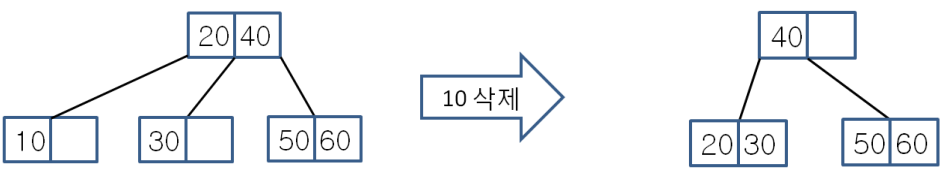


※ 개념만 맞으면 1.5, 약간 틀리면 3점

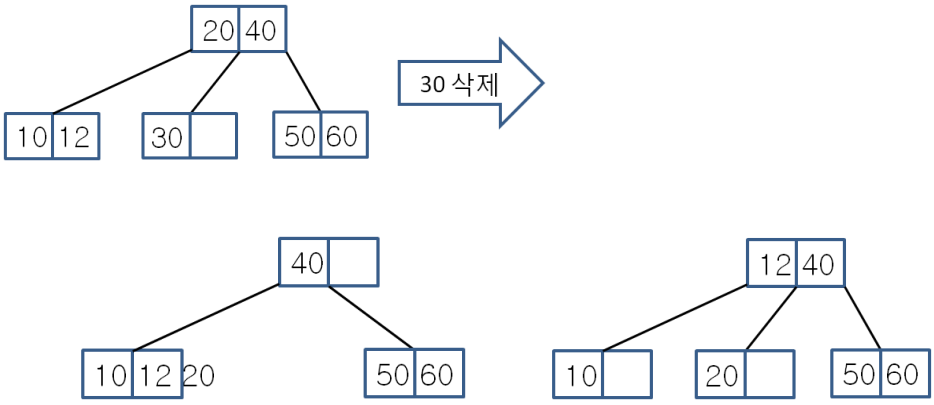
6. (5점) 2-3 트리에서 삽입 과정을 2단계로 그리시오.

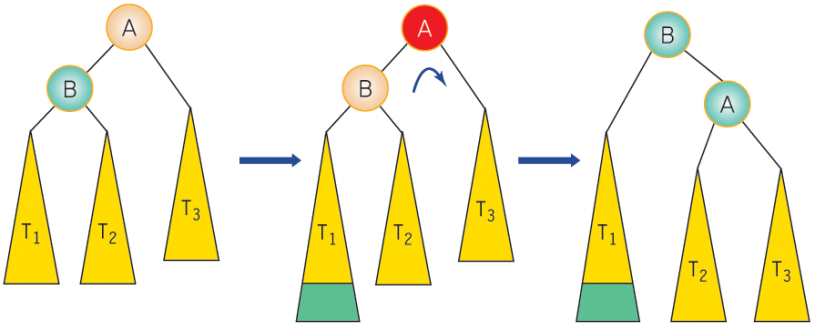


7. (5점) 2-3 트리에서 삭제 되면 어떻게 되는지 그리시오.



8. (5점) 2-3 트리에서 삭제 되면 어떻게 되는지 2단계로 그리시오.



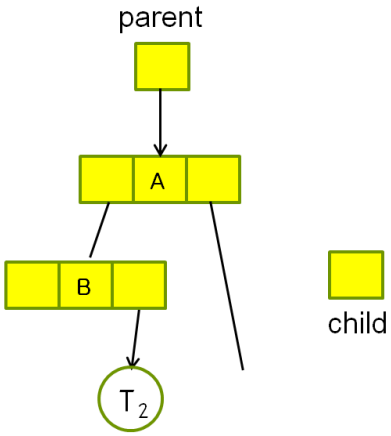


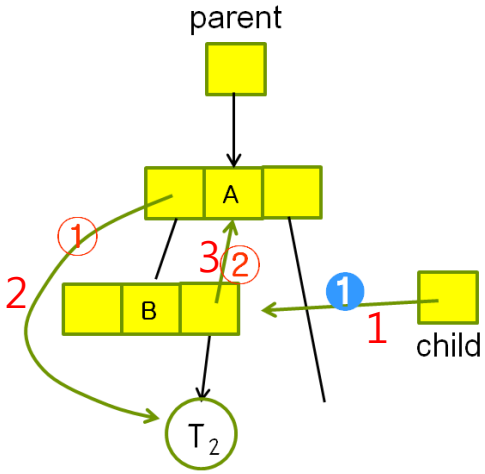

```
struct avl_node    {
        struct avl_node *left_child, *right_child;   /* Subtrees. */
        int data;                          /* Pointer to data. */
};
struct avl_node *root;
// 오른쪽 회전 함수
struct avl_node *
rotate_right(struct avl_node *parent) // 입력: parent 출력: child
{
        struct avl_node *child = parent->left_child;
        parent->left_child = child->right_child;
        child->right_child = parent;
        return child;
}
```

9. (5점) (아래)에 있는 코드의 순서를 바르게 하여 코드 안에 쓰시오.

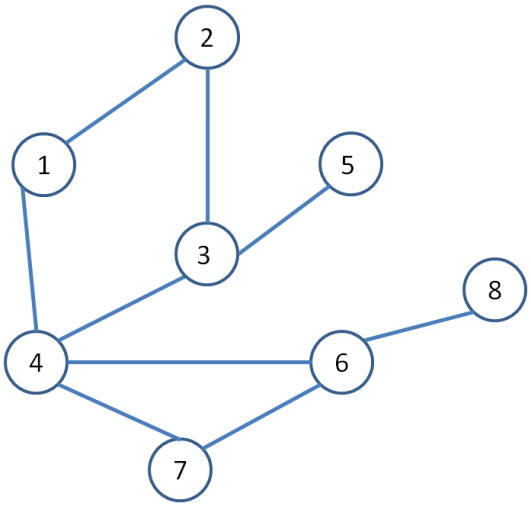
```
child->right_child = parent;
struct avl_node *child = parent->left_child;
parent->left_child = child->right_child;
```

(3) (5점) 회전하기 전의 그림은 다음과 같다. 회전후의 화살표 연결 모습을 아래 그림 위에 나타내시오. 연결 순서 1) 2) 3) 을 화살표 위에 쓰시오.





10.



(1) (5점) 깊이 우선 탐색을 하는 경우 방문 하는 정점을 차례로 쓰시오.
4-1-2-3-5-6-7-8, 1-2-3-4-6-7-8-5

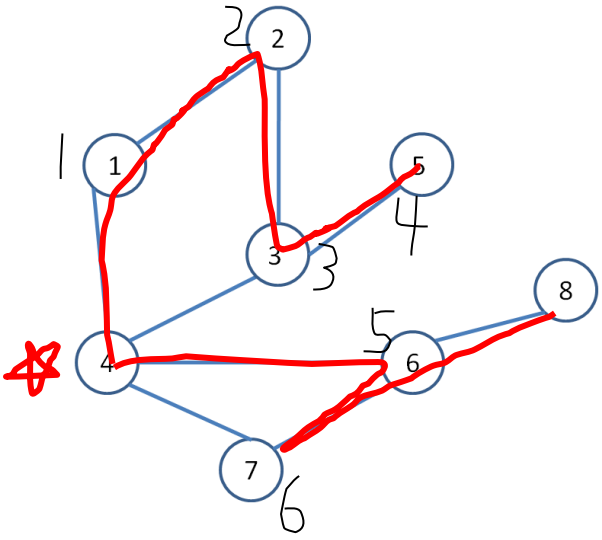
(2) (5점) 깊이 우선 탐색을 하여 만든 신장 트리를 그리시오.

(3) (5점) 너비 우선 탐색을 하는 경우 방문 하는 정점을 차례로 쓰시오.
4-1-3-6-7-2-5-8 1-2-4-3-5-6-7-8

(4) (5점) 너비 우선 탐색을 하여 만든 신장 트리를 그리시오.

※ 방문할 정점의 우선순위가 동일할 경우 1, 2, 3 오름 차순으로 방문하라.

(2)



노드 6 방문 후에 노드 7과 노드 8이 있는데, 오름 차순에 의해서 노드 7을 방문한다.

(4)