【 문제-1 】 (30점)

다음은 이진탐색트리(Binary Search Tree)에서 특정 키값을 갖는 노드를 삭제할 때 발생하는 3가지 상황(case1, case2, case3)을 재귀적인(Recursive) 방법으로 해결하는 함수이다. 이 함수는 이진탐색트리에서 키값(key)을 갖는 노드를 삭제한 후 갱신된 트리의 루트 노드 포인터(tree)를 반환한다. 다음 물음에 답하시오. (단, free()는 할당된 노드 공간을 해제하는 함수이다.)

```
typedef struct node * node_ptr;
struct node {
    int key;
    node_ptr left;
    node_ptr right;
};

node_ptr deleteBST(node_ptr tree, int key) {
    node_ptr temp;
    if (!tree) return tree;
    if (key < tree->key)
        tree->left = _____(가-1)_____;
    else if (key > tree->key)
        tree->right =_____(가-2)_____;
    else {
        // case1 and case2
        if (!tree->left) {
            _____(나-1)_____;
            free(tree);
            return temp;
        }
        else if (!tree->right) {
            _____(나-2)_____;
            free(tree);
            return temp;
        }
        // case3
        temp = findNode(_____(다-1)_____);
        tree->key = temp->key;
        _____(다-2)_____;
    }
    return tree;
}

node_ptr findNode(node_ptr tree) {
    node_ptr imsi = tree;
    while (imsi && _____(라-1)_____) {
        _____(라-2)_____;
    }
    return imsi;
}
```

(1) 이진탐색트리에서 노드 삭제시 트리의 형태별로 발생할 수 있는 3가지 상황(case1, case2, case3)과 해결 방법을 각각 설명하시오. (단, case3은 2가지 해결 방법을 모두 설명해야 한다.) (8점)

(case1)

(case2)

(case3)

(2) 문항(1)에서 설명한 (case1)과 (case2) 상황을 재귀적으로 해결하는 코드를 빈칸 (가-1), (가-2), (나-1), (나-2)에 C언어로 작성하시오. (10점)

(가-1)

(가-2)

(나-1)

(나-2)

(3) 문항(1)에서 설명한 (case3) 상황을 해결하는 방법을 빈칸 (다-1) ~ (라-2)에 C언어로 작성하시오. (단, 2가지 방법을 각각 작성해야 한다.) (12점)

(방법1)

(다-1)

(다-2)

(라-1)

(라-2)

(방법2)

(다-1)

(다-2)

(라-1)

(라-2)

【 문제-2 】(20점)

다음 표는 방향 그래프(Directed Graph)의 간선(Edge)과 비용(Cost)에 관한 정보이다. 다음 물음에 답하시오.

간선	비용	간선	비용
<a, b>	3	<a, d>	8
<b, c>	6	<c, e>	4
<c, f>	7	<d, c>	9
<d, f>	2	<e, d>	5
<e, f>	10		

(1) 이 그래프를 노드(a)에서 출발하여 깊이 우선 탐색(Depth First Search) 방법으로 탐색할 경우, 방문 노드 순서와 방문에 사용된 간선으로 구성되는 신장트리(DFS Spanning Tree)를 작성하고, 간선 비용의 합을 구하시오. (단, 간선 비용이 적은 노드를 우선 탐색한다.) (8점)

(2) 이 그래프를 노드(a)에서 출발하여 너비 우선 탐색(Breadth First Search) 방법으로 탐색할 경우, 방문 노드 순서와 방문에 사용된 간선으로 구성되는 신장트리(BFS Spanning Tree)를 작성하고, 간선 비용의 합을 구하시오. (단, 간선 비용이 적은 노드를 우선 탐색한다.) (8점)

(3) 이 그래프에 대한 비용 인접 행렬(Cost Adjacency Matrix)을 작성하시오. (4점)

【 문제-3 】 (30점)

빠른 탐색을 하기 위한 다원 탐색 트리(m-way Search Tree)에는 B-트리와 B^+-트리가 존재한다. 이 중 B-트리에 관한 다음 물음에 답하시오.

(1) 차수가 m인 B-트리의 특성 중 3가지를 설명하시오. (6점)

(2) 다음과 같은 데이터들이 순서대로 입력될 때, 차수가 3인 B-트리를 구성하는 과정을 각 데이터별로 나타내시오. (16점)
데이터 삽입 순서: 12, 3, 9, 5, 10, 15, 14, 7

(3) 문항(2)에서 구축한 B-트리에서 다음 데이터들이 순서대로 삭제될 때, 각 데이터 삭제 후 B-트리의 구조를 데이터별로 나타내시오. (8점)
데이터 삭제 순서: 10, 7

【 문제-4 】 (20점)

컴파일러는 스택을 이용하여 프로그래머가 작성한 중위표기(Infix Notation) 수식을 후위표기(Postfix Notation) 수식으로 변환하여 계산을 수행한다. 다음 물음에 답하시오. (단, 스택은 오른쪽으로 커지고, +, -, *, /는 산술연산자이고 eos는 문자열의 끝이다.)

(1) 다음 표는 중위표기 수식 a/(b-c*d)*e+f 를 후위표기 수식으로 변환하는 과정을 보여주고 있다. 토큰(연산자 또는 피연산자) 일부를 처리한 결과를 참고하여 나머지 토큰들을 처리할 때 스택과 출력결과를 나타내시오. (10점)

다음 토큰	스택					출력(후위표기 수식)
	[0]	[1]	[2]	[3]	[4]	
없음	공백 (empty)					없음
a						a
/	/					a
(	/	(				a
b						
-						
c						
*						
d						
)						
*						
e						
+						
f						
eos						

(2) 후위표기 수식 abc+d**e/의 계산 과정에서 토큰 별로 처리할 때 스택을 나타내시오. (6점)

다음 토큰	스택				
	[0]	[1]	[2]	[3]	[4]
없음	공백 (empty)				
a	a				
b	a	b			
c	a	b	c		
+					
d					
*					
*					
e					
/					
eos	공백 (스택에 있는 계산 결과 출력)				

(3) 컴파일러가 중위표기 수식을 후위표기 수식으로 변환하여 계산하는 이유를 설명하시오. (4점)