CSE 2017 자료구조와 실습

설계문제 #2: 트리를 이용한 회사 조직도

단원	트리	난이도	초급 /중급 /고급
참여인원	1명 /2주 (man/month)	마감	2017년 5월 29일(월) 12:00 AM

1. 과제 개요

회사의 조직 구조는 일반적인 트리로 표현한다. 회사의 조직은 CEO가 조직의 루트 노드이며 채용(hire)과 퇴직(fire)이 반복된다. 어떤 직원이 채용한 사원은 직속 부하 직원이 된다. 즉 트리의 노드는 직원의 이름이며 그 직원이 채용(hire)한 부하 직원이 자식 노드가 된다. 그림 1과 같은 트리가 회사 조직도의 예이다.

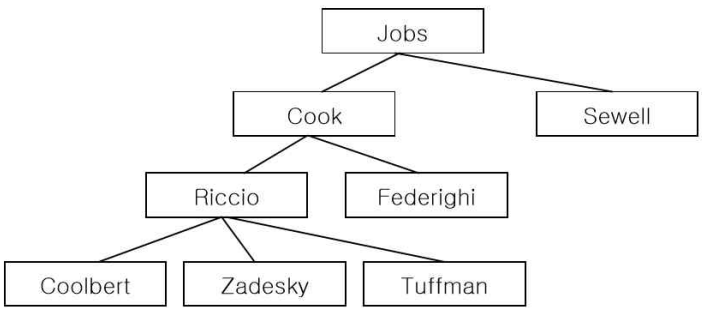


그림 1 조직의 계층 구조도

그림 1 조직도의 Jobs는 CEO이다. Jobs가 고용(hire)한 Cook과 Sewell는 직속 부하가 된다. 같은 부하 중에는 왼쪽부터 오른쪽으로 역할의 중요도에 대한 서열이 있다. 예를 들어 Cook은 Sewell보다 서열에 높다. 어떤 직원이 새로 신입 직원을 채용하였다면 신입 직원은 직속 부하 중 가장 서열이 낮다. 예를 들어 그림 1 조직에서 Jobs가 Schiller를 채용한다면 그림 2와 같이 Cook, Sewell, Schiller 순으로 서열이 정해진다.

어떤 직원이 퇴직(fire)한다면 두 가지 경우로 조직이 바뀐다. 퇴직하는 직원의 직속 부하가 없다면 그 직원만 조직에서 삭제한다. 부하 직원이 있다면 가장 서열이 높은 직원이 퇴직 직원의 서열을 이어받으며 승진한다. 예를 들어 그림 1에서 Cook이 퇴직한다면 그림 2와 같이 Rccio가 그 자리에 승진하고 Coolbert가 Riccio의 자리를 이어받아 승진한다.

그림 2는 Jobs가 Schiller를 고용(hire)하고 Cook이 퇴사한 후의 조직도이다.

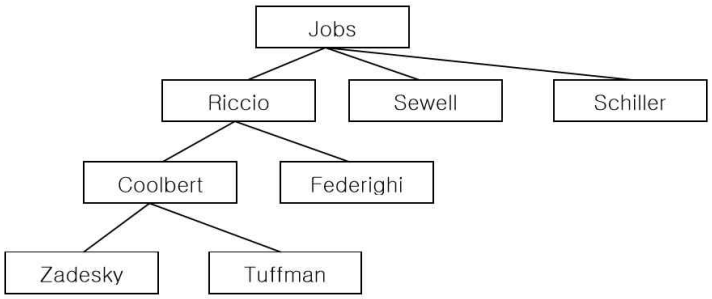


그림 2 인사 변동 후의 조직도

2. 과제 목표 및 주요 내용

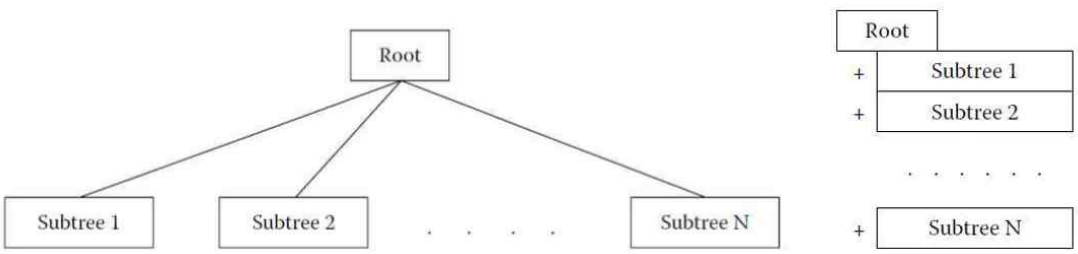
입력:
입력의 첫 번째 줄은 CEO 이름이다. 모든 이름은 3자 이상 20자 이하의 길이를 가져야 한다.
첫 문장 이후에는 다음과 같은 세 가지 문법의 문장이 올 수 있다.

[현재 조직도에 있는 직원 이름] hire [새 직원 이름]
Fire [현재 조직도에 있는 직원 이름]
Print

세 가지 문장은 순서에 관계없이 계속 입력될 수 있다. 적어도 하나의 직원(CEO)은 있어야 하며 최대 1000명의 직원이 조직도에 표현할 수 있어야 한다.

출력:
Print 문장이 입력될 때마다 조직도의 계층 구조를 다음과 같은 형태로 인쇄하여야 한다.

출력 라인은 트리에 존재하는 이름 하나씩 인쇄한다.
첫째줄은 CEO 이름을 1번째 열부터 인쇄한다.
일반적인 트리를 텍스트 형태로 인쇄하는 규칙은 그림 3과 같다.



(a) 트리 (b) 텍스트 형태
그림 3 트리를 텍스트 형태로 인쇄하는 방법

입력 예:

```
Jobs
Jobs hire Cook
Jobs hire Sewell
Cook hire Riccio
Cook hire Federighi
Riccio hire Coolbert
Riccio hire Zadesky
Riccio hire Tuffman
Print
Jobs hire Schiller
Fire Cook
Print
```

출력 예:

```
Jobs
+Cook
++Riccio
+++Coolbert
+++Zadesky
+++Tuffman
++Federighi
+Sewell
---------------------------------
Jobs
+Riccio
++Coolbert
+++Zadesky
+++Tuffman
++Federighi
+Sewell
+Schiller
--------------------------------
```

3. 자료 구조의 설계

일반적인 트리는 Left-child right-sibling 형태로 구현한다. 트리 노드와 hire의 구현은 다음을 참고하라.

```
typedef struct treeNode {
    string              name;
    struct treeNode    *child;        // left-most child
    struct treeNode    *sibling;      // 같은 parent를 가진 형제 노드
} treeNode;

void addChild(treeNode *root, treeNode *newChild) {
    treeNode *temp;
    if (root->child == NULL) {
        root->child = newChild;
    } else {
        temp = root->child;
        while (temp->sibling != NULL) {
            temp = temp->sibling;
```

```
        }
        temp->sibling = newChild}
    }
}
```