2018 학년도 2 학기 기말고사 (1/2)		과목명	자료구조 (분반: 00□)	학 과		학년		점수
담당교수	심 정 섭			학 번				
시험일시	12월 14일 금요일			성 명				

1. 다음 설명들 중 옳은 설명을 모두 고르시오. 단 n은 입력 데이터의 수이다.

① 선택정렬(selection sort) 알고리즘, 삽입정렬(insertion sort) 알고리즘, 힙정렬(heapsort) 알고리즘은 모두 제자리(in-place) 정렬 구현이 가능하다.
② 힙정렬 알고리즘은 크게 2단계로 구성되어 있으며, 1단계는 $O(n)$ 시간에, 2단계는 $O(n \log n)$ 시간에 수행되도록 구현할 수 있다.
③ 최악의 경우에 대해, 선택정렬 알고리즘과 삽입정렬 알고리즘의 입력된 데이터들에 대한 대소비교 연산 수행 횟수는 같다. (O-표기법 아님)
④ 최선의 경우에 대해, 선택정렬 알고리즘과 삽입정렬 알고리즘의 입력된 데이터들에 대한 대소비교 연산 수행 횟수는 같다. (O-표기법 아님)
⑤ 평균적인 경우(입력에 대한 동일한 확률 가정)에 대해, 선택정렬 알고리즘과 삽입정렬 알고리즘의 입력된 데이터들에 대한 대소비교 연산 수행 횟수는 같다. (O-표기법 아님)

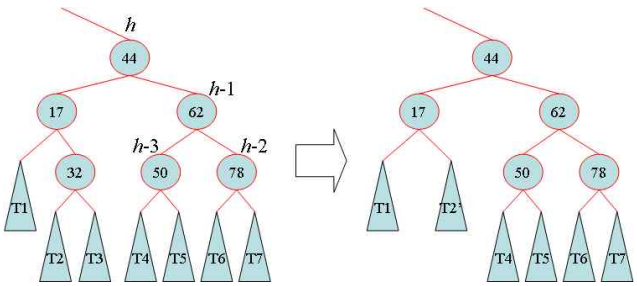
답:

2. 최소힙(min-heap)은 기본적으로 임의의 엔트리(entry)에 대한 삽입연산, 최소키를 가진 엔트리에 대한 탐색연산, 최소키를 가진 엔트리에 대한 삭제연산을 수행할 수 있다. 이를 확장하여, 하나의 엔트리에 대한 키 값을 감소시켜 우선순위를 높여주는 함수 decrease_key를 설계하려 한다. decrease_key(p, i)는 엔트리의 위치(position)가 p일 때, 해당 엔트리의 키값 $p.key$를 정수 i로 감소시킨다. 이를 C 또는 C++ 언어로 구현하려 할 때, 힙은 배열로 구현되는데 엔트리에 대한 위치를 이용하여 decrease_key 함수가 호출되므로 ① 자료구조(힙과 엔트리의 클래스 또는 구조체) 설계시 고려해야 할 사항 설명하시오. 또한 ② decrease_key 알고리즘 및 복잡도에 대해 설명하시오.

3. ① 오픈어드레싱(Open Addressing) 기법의 예를 하나 들고, 예를 든 방법에서 삽입, 삭제 연산을 수행하기 위해 ② 유의해야 할 점과 ③ 해결방법을 설명하시오.
①:
②:

③:

4. 아래의 왼쪽 그림은 AVL 트리이고 노드 안의 수는 key 값이며 노드 밖의 수는 해당 노드가 루트노드인 부분트리(subtree)의 높이(height)이다. 이때, key가 32인 자료를 지우고 난 뒤 트리의 모양이 오른쪽 그림과 같이 바뀌었다. 그림에서 key가 44인 노드가 AVL 트리의 특성에 문제를 일으키는 첫 번째 노드라 할 때, AVL 트리의 특성을 만족하도록 변화되는 과정을 설명하고 결과를 그림으로 표현하시오. 단, 그림에는 아래 각 다섯 개의 노드가 루트노드인 부분트리들의 높이를 표시하시오.



5. 자료구조 강의를 수강하는 n명의 학생들 사이의 연락처 보유 현황을 그래프 G로 모델링하려한다. 이때 학생 A가 학생 B의 연락처를 알지만 B가 A의 연락처를 모를 수도 있다. 구성원들을 정점으로 모델링하고, 학생 A가 학생 B의 연락처를 알면 A의 정점에서 B의 정점으로 간선을 생성한다. $G=(V,E)$(단, $|V|=n, |E|=m$)를 강의시간에 설명한 ①인접리스트(adjacency list)와 ②인접행렬(adjacency matrix)로 각각 표현하였다. 다음 빈 칸을 적절히 채우시오.

(1) ②를 이용했을 때 G를 표현하기 위해 필요한 공간

답: $O($ $)$

(2) ①을 이용했을 때 임의의 학생 A가 임의의 학생 B에게 연락할 수 있는지 여부 확인하는 최악수행시간

답: $O($ $)$

(3) ①을 이용했을 때 임의의 학생 A가 임의의 학생 B에게 최소 몇 번 만에 연락할 수 있는지 확인하는 최악수행시간

답: $O($ $)$

(4) ②를 이용했을 때 임의의 학생 A가 임의의 학생 B의 연락처를 새롭게 알게 됐을 때 이 정보를 추가하는 최악수행시간

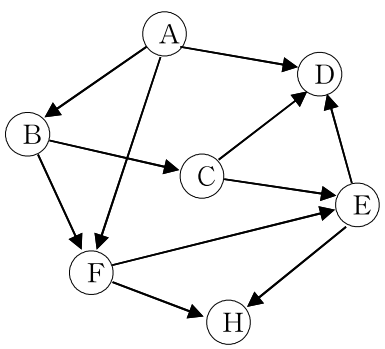
답: $O($ $)$

6. 아래 두 알고리즘은 모두 정점의 수가 n인 유향그래프(digraph) G에 대한 위상정렬을 수행한다. G가 아래 그림과 같이 주어졌을 때, 다음 물음들에 답하시오.

```
Algorithm TopologicalSort(G)
    H ← G
    i ← n
    while H is not empty do
        Let v be a vertex with no outgoing edges
        Label v ← i
        i ← i – 1
        Remove v from H
```

```
Algorithm TopologicalSort2(G)
    S ← an initially empty stack
    for all u in G.vertices() do
        Let incounter(u) be the in-degree of u.
        if incounter(u) = 0 then S.push(u)
    i ← 1
    while !S.empty() do
        u ← S.pop()
        Let u be vertex number i in the topological ordering.
        i ← i + 1
        for all outgoing edges (u,w) of u do
            incounter(w) ← incounter(w) - 1
            if incounter(w) = 0 then S.push(w)
```

G:



(1) 알고리즘 *TopologicalSort*를 강의시간에 학습한 깊이우선탐색(DFS) 알고리즘을 이용하여 구현하려 한다. 아래 주어진 알고리즘 *TopologicalDFS*(G)를 참조하여 *TopologicalDFS*(G,v)의 의사코드를 작성하시오.

```
Algorithm TopologicalDFS(G)
    Input DAG G
    Output topological ordering of G
    for all u in G.vertices()
        u.setLabel(UNEXPLORED)
    for all v in G.vertices()
        if v.getLabel() = UNEXPLORED
            TopologicalDFS(G, v)
```

```
Algorithm topologicalDFS(G, v)
    Input graph G and a start vertex v of G
    Output labeling of the vertices of G
           in the connected component of v
```

(2) G에 대해 *TopologicalDFS*를 수행했을 때 각 정점들에 붙여지는 번호를 쓰시오. 단, 여러 정점이나 간선이 리턴될 수 있을 때, 강의시간에 설명한 대로 정점의 경우 해당정점이, 간선의 경우 해당정점의 반대편 정점이 사전순으로 빠른 객체가 먼저 리턴된다고 가정한다.

답:

정점	A	B	C	D	E	F	H
번호							

(3) G에 대해 *TopologicalSort2*를 수행했을 때 각 정점들에 붙여지는 번호를 쓰시오. 단, 정점과 간선이 리턴되는 순서의 가정은 (2)와 같다.

답:

정점	A	B	C	D	E	F	H
번호							

(계산용 여백, 이 부분은 채점하지 않음)