2016 학년도 2 학기 기말고사 (1/2)		과목명	자료구조 (분반: 00□)	학 과		학년		점수
담당교수	심 정 섭			학 번				
시험일시	12월 9일 금요일			성 명				

1. 아래 의사코드(pseudo code)는 우선순위큐 추상자료형(priority queue ADT)을 이용하여 데이터를 정렬하는 알고리즘이다. 입력시퀀스 S가 크기 n인 배열로 주어졌을 때, 다음 물음들에 답하시오.

```
Algorithm PQ-Sort(S, C)
  Input sequence S, comparator C for the elements of S
  Output sequence S sorted in increasing order according to C
  P ← priority queue with comparator C
  while ¬S.empty()
      e ← S.front(); S.eraseFront()
      P.insert (e, ∅ )
  while (1)
```

(1) 의사코드에서 첫 번째 while 문을 참고하여, 두 번째 while 문을 완성하시오.

```
while
```

(2) 우선순위큐를 최소힙(min-heap)으로 구현했을 때, 첫 번째 while 문의 기능을 강의시간에 설명(bottom-up)한 대로 $O(n)$ 시간에 수행하도록 구현하였다. 배열로 주어진 S의 초기 입력이 아래와 같을 때, 최종 (배열)힙의 상태를 나타내시오.

* 초기 입력

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	16	4	11	3	12	9	5	6	14	2	7	10	1	15	13

답: 최종 힙의 내용

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-															

(3) (2)번의 방법이 $O(n)$ 시간에 수행됨을 보이시오.

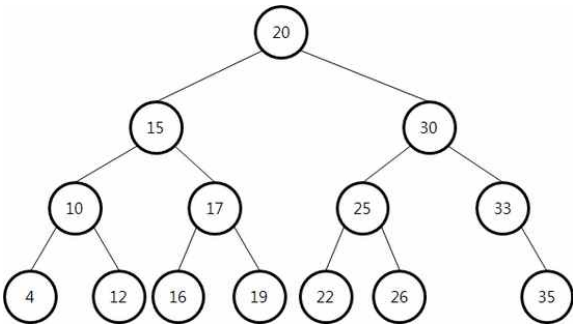
2. 다음 빈 칸들을 적절히 채우시오.

(1) A ( ) is a dictionary implemented by means of an unsorted sequence.

(2) Let α denote the load factor. Assuming that the hash values are like random numbers, it can be shown that the expected number of probes for an insertion with open addressing is ( ).

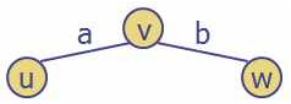
(3) Algorithm upheap restores the ( ) property by swapping the inserted key k along an upward path from the insertion node.

3. 아래 이진탐색트리(binary search tree)에서 20에 대한 삭제연산이 수행된 후, 트리의 상태를 그림으로 나타내시오. 단, 외부노드는 그림에서 생략한다.



답:

4. 아래 그래프에 대해 다음 물음에 답하시오.



(1) 강의시간에 학습한 인접리스트(adjacency list) 표현 방법을 이용하여 위 그래프를 구현하려 한다. 구현된 그래프를 그림으로 나타내시오.

(2) 정점(vertex) 객체에 정의되어 있는 데이터들이 무엇인지 설명하시오.

(3) 정점 v를 삭제하는 함수의 수행과정을 설명하고, 수행시간을 분석하시오.

5. 다음 중 그래프의 입력크기에 비례하는 시간에 계산할 수 있는 경우를 모두 고르시오.

① 모든 간선의 가중치가 1일 때, 임의의 두 정점사이의 최단거리
② DAG(directed acyclic graph)의 위상순서(topological order)
③ DAG의 한 정점에서 다른 정점까지의 최단거리
④ 무향그래프(undirected graph) G의 연결성분(connected component)

답:

6. 아래 의사코드에서 빈 칸을 적절히 채우고, 이를 인접리스트로 구현된 $G=(V,E)$에 대해 수행했을 때, 수행시간을 설명하시오. 단, $|V|=n$, $|E|=m$이다.

```
Algorithm AAA(G)
    Input graph G
    Output labeling of the edges of G
           as discovery edges and back edges
    for all u ∈ G.vertices()
           u.setLabel(UNEXPLORED)
    for all e ∈ G.edges()
           e.setLabel(UNEXPLORED)
    for all v ∈ G.vertices()
        if v.getLabel() = UNEXPLORED
                AAA(G, v)
```

```
Algorithm AAA(G, v)
    Input graph G and a start vertex v of G
    Output labeling of the edges of G
           in the connected component of v
           as discovery edges and back edges
    v.setLabel(VISITED)
    for all e ∈ G.incidentEdges(v)
        if e.getLabel() = UNEXPLORED
           w ← e.opposite(v)
           if w.getLabel() = UNEXPLORED
              e.setLabel(DISCOVERY)
              (          ①          )
           else
              e.setLabel(BACK)
```

① :

수행시간: