1. 배열로 구현한 최대 히프(max heap)에 숫자 3, 4, 5, 6, 7, 8, 9를 순서대로 삽입하였다. 이 최대 히프에서 루트에 존재하는 값을 삭제하는 연산을 2회 수행하였을 때, 배열의 마지막에 위치한 숫자는?

① 3
② 4
③ 5
④ 6

2. 5개의 버킷(bucket)이 있고, 버킷당 2개의 슬롯(slot)을 가지는 해시 테이블과 여기에 해싱을 위한 해시 함수 hash_function()이 <보기 2>에 있다. 버킷 번호는 0, 1, 2, 3, 4로 설정하며, 해시 함수 hash_function()은 탐색키가 문자열인 경우에 해시값을 반환한다. 문자열 데이터가 <보기 1> 순서대로 해싱 과정이 수행될 때, 처음으로 오버플로가 발생하는 버킷 번호는?

<보기 1>

"array", "data", "graph", "heap", "list", "buffer", "circle", "bit"

<보기 2>

```
int hash_function(char key[])
{
    int diff=key[0]-'a';
    int hash=diff%5;
    return hash;
}
```

① 0
② 1
③ 2
④ 3

3. 그래프(graph)와 관련된 설명 중 가장 옳지 않은 것은?

① 무방향 그래프(undirected graph)를 인접 행렬로 표현하면 대칭행렬이 된다.
② 15개의 정점(vertex), 20개의 간선(edge)으로 이루어진 무방향 그래프를 인접 리스트(adjacency list)로 표현할 때, 인접 리스트에 존재하는 노드의 총 개수는 30이다.
③ 20개의 정점으로 구성된 무방향 완전 그래프(undirected complete graph)에 존재하는 간선의 총 개수는 190이다.
④ 100개의 정점으로 구성된 그래프에 대한 신장 트리(spanning tree)는 99개의 간선으로 연결된다.

4. <보기>와 같은 함수에서 f(4)를 수행한 결과 값은?

<보기>

```
int f(int n)
{
    if (n<=1) return n;
    return (5*f(n-1)-6*f(n-2));
}
```

① 9
② 19
③ 56
④ 65

5. <보기>의 C언어 2차원 배열에 대한 설명으로 가장 옳지 않은 것은?

<보기>

```
int mat[3][4]={1, 2, 3, 4, 5, 6};
```

① mat[2][2]의 값은 0이다.
② mat[1]==&mat[1][0]이고, mat[1]의 자료형은 int*이다.
③ **mat은 mat[0][0]과 같은 의미이고, 자료형은 int*이다.
④ *(mat[2]+3)은 mat[2][3]을 의미한다.

6. 함수 func1(), func2(), func3(), func4()의 시간 복잡도를 빅오(Big-O)표기법으로 표현한 것으로서 옳지 않은 것은?

①

```
void func1(int n)
{
    int i, j;
    for (i=0; i<n; i++)
      for (j=i; j<n; j++)
        printf("%d %d\n", i, j);
}
```

$O(n^2)$

②

```
void func2(int n)
{
    int i;
    for (i=1; i<=n; i=i*2)
        printf("%d\n", i);
}
```

$O(\log_2 n)$

③

```
int func3(int n)
{
    if (n<=1) return 1;
    else return (n*func3(n-1));
}
```

$O(n)$

④

```
int func4(int n)
{
    int k=0;
    while (n>2){
      n=n/2; k++;
    }
    return k;
}
```

$O(n \cdot \log_2 n)$

7. 서로 다른 5개의 값을 저장하는 이진 탐색 트리로 구성 가능한 개수는?
① 14개
② 21개
③ 42개
④ 128개

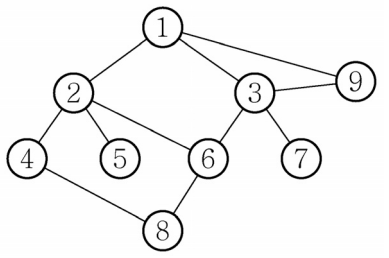
8. 설명 중 가장 옳지 않은 것은?
① 자료 구조는 자료 객체의 집합뿐 아니라 그들의 관계, 즉 자료 객체의 원소에 합법적으로 적용될 연산들까지 기술하는 것이다.
② 시간 복잡도(time complexity)를 측정하는 방법으로 알고리즘의 기본 연산 수행 횟수를 세는 방법이 있다.
③ 알고리즘이 수행될 때 소요되는 메모리 양을 분석한 것을 공간 복잡도(space complexity)라고 한다.
④ 빅오(big-O) 표기법은 시간 복잡도를 나타내고, 오메가(omega, Ω) 표기법은 공간 복잡도를 나타낸다.

9. 스택(stack)과 큐(queue)에 대한 설명으로 가장 옳지 않은 것은?
① 스택은 제일 먼저 삽입된 원소가 제일 나중에 삭제된다.
② 큐의 응용 분야로 작업 스케줄링(job scheduling)이 적합하다.
③ 스택의 응용 분야 중 하나는 서브루틴 호출(subroutine call)이다.
④ 스택의 추상 데이터 타입은 1차원 배열로 구현할 수 있으나, 큐의 경우는 불가능하다.

10. 원형 연결 리스트(circular linked list)의 각 노드는 DATA 필드와 LINK 필드로 구성된다. DATA 필드는 데이터 값을 저장하고, LINK 필드는 포인터(pointer) 값을 저장한다. 첫 번째 노드를 가리키는 포인터를 head, 마지막 노드를 가리키는 포인터를 tail이라고 하면, 이 리스트의 첫 번째 노드의 주소를 가리키는 값은?
① tail → LINK
② tail → DATA
③ tail → LINK → LINK
④ tail → LINK → DATA

11. <보기>의 그래프에서 깊이 우선 탐색(Depth First Search) 결과를 나타낸 것들 중 가장 옳지 않은 것은?

<보기>



① 1 – 2 – 5 – 6 – 3 – 7 – 9 – 8 – 4
② 1 – 2 – 5 – 6 – 8 – 4 – 3 – 7 – 9
③ 1 – 3 – 9 – 7 – 6 – 2 – 4 – 8 – 5
④ 1 – 3 – 9 – 6 – 2 – 5 – 8 – 4 – 7

12. <보기>의 이진 트리에 대한 함수 traverseA와 traverseB의 운행 방식을 가장 옳게 나타낸 것은?

<보기>

```
struct NODE{
    char name[10];
    struct NODE *left;
    struct NODE *right;
};
typedef struct NODE Node;
```

```
void traverseA(Node *x)
{
    if (x!=NULL){
        traverseA(x->left);
        printf("%s\n", x->name);
        traverseA(x->right);
    }
}

void traverseB(Node *x)
{
    if (x!=NULL){
        traverseB(x->left);
        traverseB(x->right);
        printf("%s\n", x->name);
    }
}
```

	traverseA	traverseB
①	중위 순회	후위 순회
②	후위 순회	중위 순회
③	중위 순회	전위 순회
④	전위 순회	후위 순회

13. 레드-블랙 트리(red-black tree)에 대한 설명으로 가장 옳지 않은 것은?
① 레드-블랙 트리에서 임의의 노드(node)의 두 자식트리(child tree)의 높이(height) 차이는 1 이하이다.
② 레드-블랙 트리에 내부 노드가 n개 있으면, 높이는 최대 $2\log_2(n+1)$이다.
③ 레드-블랙 트리의 루트 노드는 흑색(black) 노드이다.
④ 레드-블랙 트리에서 적색(red) 노드의 모든 자식 노드는 흑색 노드이다.

14. 입력 리스트를 특정한 키(key) 값보다 작은 값을 갖는 자료와 큰 값을 갖는 자료로 분리하여 한 개의 리스트를 논리적으로 두 개의 서브(sub) 리스트로 재배열한 후, 각각의 서브 리스트에 대해 순환적으로 같은 정렬 방법을 적용하여 입력 리스트를 정렬하는 방법은?

① 버블 정렬(Bubble Sort)
② 히프 정렬(Heap Sort)
③ 퀵 정렬(Quick Sort)
④ 삽입 정렬(Insertion Sort)

15. 트리(tree)에 대한 설명 중 가장 옳지 않은 것은?

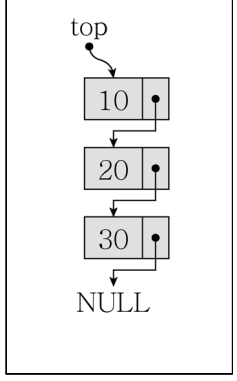
① 2-3 트리는 균형 트리(balanced tree)이다.
② 트리는 계층적인 자료를 표현하는 데 이용된다.
③ 균형 트리(balanced tree)는 불균형 트리에 비해 노드 추가가 간단하고 빠르다.
④ 경사 트리(skewed tree)는 이진 트리(binary tree)에 속하며, 배열(array) 표현법을 사용하면 기억공간의 낭비가 많아진다.

16. 1차원 배열에 저장된 히프(heap) 구조를 갖는 데이터 중에서 최대 히프(max heap)의 성질을 만족하지 않는 배열 상태는?

① 8 4 7 3 2 6 5 1
② 8 6 7 2 4 1 5 3
③ 8 7 6 5 4 1 2 3
④ 8 7 6 5 4 3 2 1

17. <보기 1>은 연결 리스트(linked list)로 구현된 스택(stack)이고, <보기 2>는 연결 리스트로 구현된 스택으로부터 데이터 항목의 개수를 반환하는 stack_size() 함수이다. 여기서 변수 top은 스택의 최상단의 노드를 가리키는 포인터 변수이다. <보기 2>의 ㉠, ㉡에 들어갈 코드로서 옳은 것은?

<보기 1>



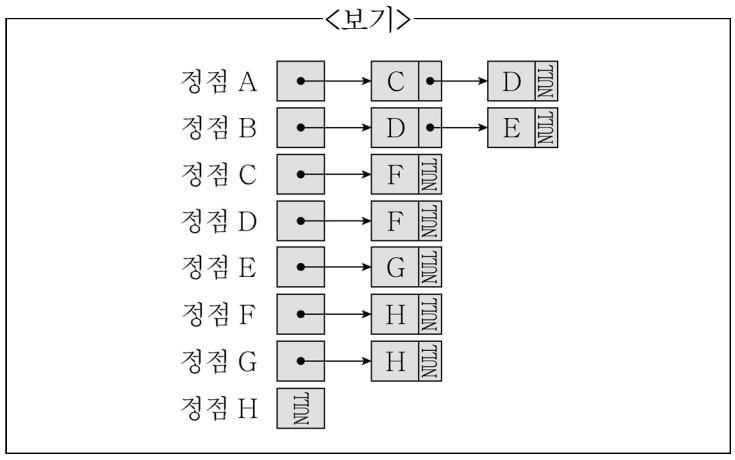
<보기 2>

```
typedef struct LinkedNode{
        int data;
        struct LinkedNode *link;
}Node;
int stack_size()
{
        Node *p;
        int count=0;
        for (p=top;[   ㉠   ];[   ㉡   ])
             count++;
        return count;
}
```

	㉠	㉡
①	p!=NULL	p=p->link
②	p!=p->link	p=p->link
③	p!=NULL	p++
④	p!=p->link	p++

18. <보기>는 인접 리스트(adjacency list)로 표현한 방향 그래프이다. 이 그래프에 대하여 위상 정렬(topological sort)을 수행한 결과로서 가장 옳지 않은 것은?

<보기>



① A, B, C, D, E, F, G, H
② A, C, B, E, G, D, F, H
③ B, E, G, A, D, C, F, H
④ B, E, G, H, D, F, A, C

19. 이진 트리(binary tree)에 대한 설명 중 가장 옳지 않은 것은?

① 100개로 구성되어 있는 완전 이진 트리(complete binary tree)에서 최하위 수준(level)에는 36개의 단말 노드가 존재한다.

② 이진 트리에서 좌/우측 자식 노드를 모두 가진 노드의 개수가 100이라면, 자식 노드가 없는 단말 노드의 개수는 101이다.

③ 이진 트리를 배열(array) 표현법으로 구현할 때, 인덱스 101번에 저장된 노드의 부모 노드 인덱스는 50번이다.

④ 이진 트리를 링크(link) 표현법으로 구현할 때, 이진 트리에 포함된 노드의 개수가 100이라면, NULL 링크의 개수는 101이다.

20. <보기 1>의 인접 행렬로 표현된 그래프 G에 <보기 2>의 알고리즘으로 최소비용신장트리(minimum cost spanning tree)를 생성할 때, 추가되는 간선의 순서로서 옳은 것은? (단, s는 시작 정점을 의미하며, 본 문항에서는 s를 0번 정점으로 한다.)

<보기 1>

	0	1	2	3	4	5	6	7	8
0	0	3	0	2	0	0	0	0	4
1	3	0	0	0	0	0	0	4	0
2	0	0	0	6	0	1	0	2	0
3	2	0	6	0	1	0	0	0	0
4	0	0	0	1	0	0	0	0	8
5	0	0	1	0	0	0	8	0	0
6	0	0	0	0	0	8	0	0	0
7	0	4	2	0	0	0	0	0	0
8	4	0	0	0	8	0	0	0	0

<보기 2>

```
MST(G, s)
{
  1. 그래프 G에서 시작 정점 s를 가지고 초기 신장 트리를 만듦
  2. 현재 트리의 정점들과 인접한 정점들 중에서 간선의 가중치가
     가장 작은 정점 v를 선택
  3. 선택된 정점 v와 간선을 신장 트리에 추가 (단, 사이클(cycle)
     이 발생하지 않아야 함)
  4. 현재 신장 트리에 모든 정점이 포함되지 않았다면 2번 단계로
     이동
}
```

① (0, 3), (3, 4), (4, 8), (3, 2), (2, 7), (7, 1), (2, 5), (5, 6)

② (0, 3), (3, 4), (0, 1), (1, 7), (7, 2), (2, 5), (0, 8), (5, 6)

③ (0, 3), (0, 1), (0, 8), (1, 7), (7, 2), (2, 3), (2, 5), (5, 6)

④ (0, 3), (3, 4), (0, 8), (0, 1), (1, 7), (7, 2), (2, 5), (5, 6)