〓자료구조 1학기 형성평가〓

1. 선형구조를 모두 나열하시오.
[정답] 연결 리스트, 스택, 큐, 배열, 테이블의 레코드

2. 비선형구조를 모두 나열하시오.
[정답] 트리, 그래프

3. 밑줄에 해당하는 자료에 대한 기본 작업을 쓰시오.
(1) 초기화 (initialization): 자료를 생성할 때 특정 값으로 설정
(2) 소멸 (destruction): 자료가 사용했던 기억장소를 다른 목적으로 재사용할 수 있게 점유를 해제함
(3) 조회 (retrieval): 자료구조 또는 데이터베이스에서 조건에 맞는 자료원소 또는 튜플들을 찾아 가져와서 이를 출력장치에 나타냄
(4) 순회 (traversal): 자료구조에서 모든 원소를 한번만 방문하면서 전체 자료를 처리
(5) 삽입 (insertion): 자료구조에 새로운 원소 삽입
(6) 삭제 (deletion): 자료구조에서 기존의 원소 삭제. 기억장소가 유지된다는 점에서 소멸과 차이가 있음
(7) 탐색 (searching): 자료구조에서 목표 원소 또는 탐색키를 갖고 있는 자료를 찾는 작업
(8) 정렬 (sorting): 자료구조에서 자료 원소를 순서대로 나열하는 작업
(9) 합병 (merging): 두 개 이상의 자료 집합을 하나로 합치는 작업. 병합이라고도 함

4. 괄호 속에 해당하는 알고리즘 충족요건을 쓰시오.
· (유한성): 언젠가는 종료되어야 함
· (명확성): 각 단계별로 무엇을 하여야 하는지를 구체적으로 표현하여야 함
· (입력): 처리를 위하여 0개 이상 필요함
· (출력): 1개 이상의 출력을 얻기 위해 알고리즘은 존재함
· (효과성): 효과적인 방법으로 결과를 만들어 내야 함

5. 데이터 수 증가에 따라 수행 시간을 가늠할 수 있는 O 표기법을 무엇이라 하는가?
[정답] 시간복잡도, 빅 오 표기법

6. 데이터 수에 따라 주기억장치 사용량을 가늠할 수 있는 O 표기법을 무엇이라 하는가?
[정답] 공간복잡도

7. 8가지 대표적인 시간복잡도에 대하여

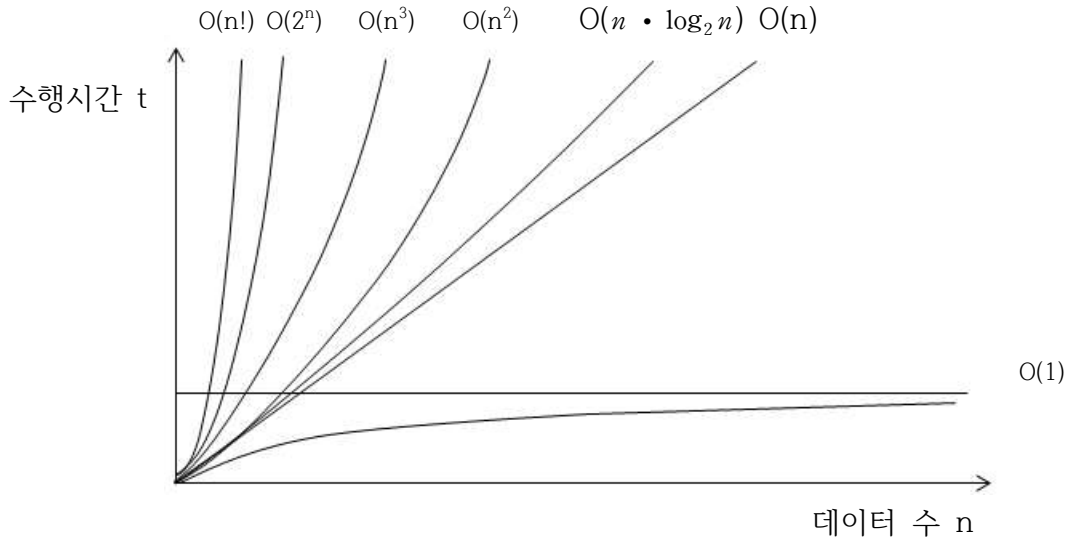
가. 다음 빈 칸을 채우시오.

종 류	설 명
O(1)	상수형. 데이터 수에 관계없음
$O(\log_2 n)$	로그형. 데이터 수에 따라 조금씩 늘어남
O(n)	선형. 데이터 수에 따라 산술급수적으로 증가함
$O(n \cdot \log_2 n)$	선형로그형. 로그형과 선형의 곱
$O(n^2)$	평방형. 데이터 수에 따라 기하급수적으로 증가함
$O(n^3)$	입방형. 데이터 수에 따라 기하급수적으로 증가함
$O(2^n)$	지수형. 데이터 수에 따라 기하급수적으로 증가함
$O(n!)$	계승형. 데이터 수에 따라 기하급수적으로 증가함

나. 수행 시간이 짧은 시간 복잡도부터 나열하시오.

$O(1) < O(\log_2 n) < O(n) < O(n \cdot \log_2 n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$

다. 그래프 계열 이름을 쓰시오.



8. 다음 코드에 대하여

```
#include <stdio.h>

void HanoiTower(int n, char a, char b, char c) {
        if (n == 1) printf("원판 %d, %c -> %c\n", n, a, c);
        else {
                HanoiTower(n - 1, a, c, b);
                printf("원판 %d, %c -> %c\n", n, a, c);
                HanoiTower(n - 1, b, a, c);
        }
}

int main(void) {
        int n = 5;
        HanoiTower(n, 'A', 'B', 'C');

        return 0;
}
```

가. 출력 결과를 10줄까지 정확히 쓰시오.

원판 1, A->C
원판 2, A->B
원판 1, C->B
원판 3, A->C
원판 1, B->A
원판 2, B->C
원판 1, A->C
원판 4, A->B
원판 1, C->B
원판 2, C->A

나. 시간 복잡도(Time Complexity), 즉 Big O notation(빅 오 표기법)을 쓰시오.

$O(2^n)$

9. 배열과 연결리스트 비교표에서 괄호 부분을 완성하시오.

구분	배열	연결리스트
검색	데이터에 (직접)접근하므로 읽기/쓰기가 연결리스트에 비해 빠름	데이터에 (간접) 접근해야 하므로 읽기/쓰기가 배열에 비해 느림
데이터/구조 변경	데이터 추가/삭제 시 반복 이동이 필요함	노드 추가/삭제 용이
기억 공간	코딩 시 정해진 크기만큼 (정적) 할당	노드 필요 시 (동적) 할당하여 연결함
추가 기억 공간	추가 공간 없음	링크 필드 추가
선형 구조	(O)	(O)
소멸	지역 배열은 (함수) 종료 때 전역 배열은 프로그램 종료 때	매 노드마다 free() 함수를 사용

10. 코드의 실행 결과를 쓰시오.

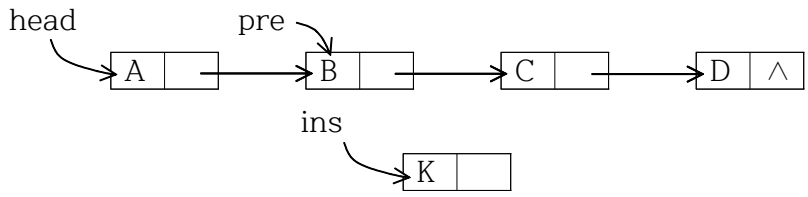
```
#include<stdio.h>

int main(void){
        int a[10] = { 3, 6, 9, 12, 15 };
        int *ptr;

        printf("출력 결과\n);
        ptr = a;
        printf("가. %d, %d\n", *ptr, *(ptr + 4));
        ptr += 3;
        // a += 3; a는 배열명이지 변수가 아님. 따라서 에러 발생
        printf("나. %d, %d, %d, %d\n", a[1], *(a + 2), ptr[0], ptr[2]);

        return 0;
}
```

가. 3, 15

나. 6, 9, 12, 0

11. 연결 리스트 노드 삽입, 삭제

가. 두 번째(pre 노드) 다음에 ins 노드를 삽입하는 문장을 완성하시오.



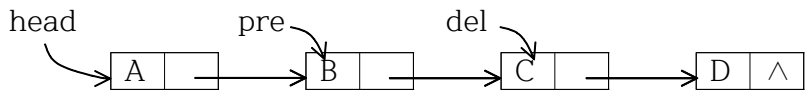

```
struct node{
    char data;
    struct node *link;
};
```

[노드 삽입 코드]

```
pre = head -> link;
```

ins = (struct node*)malloc(sizeof(struct node));
ins -> data = 'K';
ins -> link = pre -> link;
pre -> link = ins;

나. 세 번째 노드(간접 소유자는 del)를 삭제하는 문장을 완성하시오.



[노드 삭제 코드]
pre = head -> link;
del = pre -> link;
pre -> link = del -> link;
free(del);

12. <보기>는 여러 가지 자료구조의 특징을 나열한 것이다.

<보기>

(a) 선형 구조이다.o	(b) 비선형 구조이다.
(c) FIFO 구조이다.o	(d) LIFO 구조이다.o
(e) push(), pop() 동작이 있다.o	(f) add(), delete() 동작이 있다.o
(g) 'int top = -1'을 준비한다.o	(h) 부모, 자식, 형제가 존재한다.
(i) 반드시 정적 할당 공간만 사용한다.	(j) 반드시 동적 할당 공간만 사용한다.
(k) 모든 언어에서 기본적으로 제공된다.	(l) 'int rear = -1, front = 0'을 준비한다.o
(m) 배열로 구현한다.o	(n) 연결리스트로 구현한다.o
(o) 차수(degree)가 존재한다.	(p) 레벨(Level)이 존재한다.
(q) 미로 찾기에 활용된다.o	(r) Level Order Traversal에 활용된다.

가. 스택(Stack)의 특징만 고르시오.
d,e,g,m,a,n,q

나. 큐(Queue)의 특징만 고르시오.
a,c,f,i,m

13. 시스템 구축 과정에서 ()속을 채우시오.

(사용자 환경) - (시스템 정의) - (요구 분석) - (구조설계) - (상세설계) - (코딩) - (디버깅) - (단위시험) - (통합시험) - (시스템 시험) - (인수 시험) - (설치 시험)

14. 다음에 해당하는 테스트 방법을 쓰시오.

단위 시험(Test)에서 주로 사용하는 기법으로 원시 코드를 오픈시킨 상태에서 모든 논리적인 경로에 대하여 검사를 수행하는 방식을 말한다.

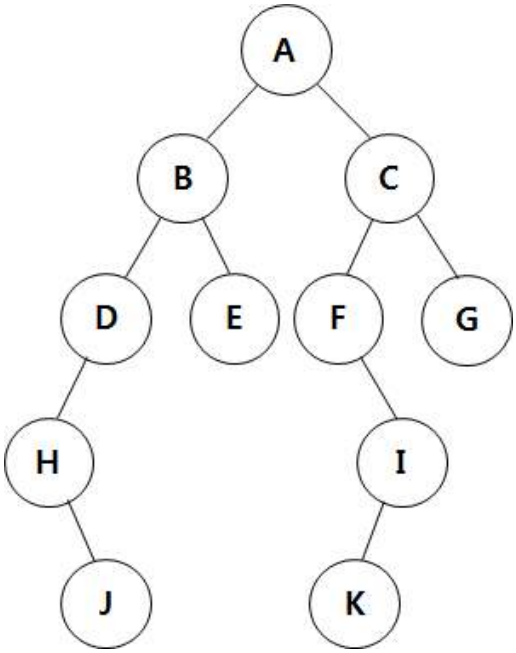
[정답]화이트 박스 테스트

15. 다음에 해당하는 테스트 방법을 쓰시오.

통합 시험(Test)에서 주로 사용하는 기법으로 내부 구조나 작동 원리를 모르는 상태에서 사용자의 요구대로 각 기능들이 정확하게 작동하는지를 확인함으로써 소프트웨어의 동작을 검사하는 방식을 말한다.

[정답]블랙 박스 테스트

16. 이진트리 및 운행 코드를 보고 운행 결과를 쓰시오.



[Preorder 운행 코드]

```
void preOrder(struct node *ptr){
        if (ptr != NULL){
                printf("%c  ", ptr->data);
                preOrder(ptr->llink);
                preOrder(ptr->rlink);
        }
}
```

[Preorder 운행 결과]

A,B,D,H,J,E,C,F,I,K,G

[Inorder 운행 코드]

```
void inOrder(struct node *ptr){
        if (ptr != NULL){
                inOrder(ptr->llink);
                printf("%c  ", ptr->data);
                inOrder(ptr->rlink);
        }
}
```

[Inorder 운행 결과]

H,J,D,B,E,A,F,K,I,C,G

[Postorder 운행 코드]

```
void postOrder(struct node *ptr){
        if (ptr != NULL){
                postOrder(ptr->llink);
                postOrder(ptr->rlink);
                printf("%c  ", ptr->data);
        }
}
```

[Postorder 운행 결과]

J,H,D,E,B,C,I,K,F,G,A

17. 이진트리 운행의 시간복잡도(time complexity), 즉 big O notation을 쓰시오.

O(n)

18. 교재 107쪽의 A 노드 주소는 ☆$_0$, B 노드 주소는 ☆$_1$, C 노드 주소는 ☆$_2$, D 노드 주소는 ☆$_3$, E 노드 주소는 ☆$_4$, F 노드 주소는 ☆$_5$, G 노드 주소는 ☆$_6$, H 노드 주소는 ☆$_7$일 때,

가. inOrder 운행의 과정 추적에서 다음 단계를 완성하시오.

*inOrder 운행(미완)

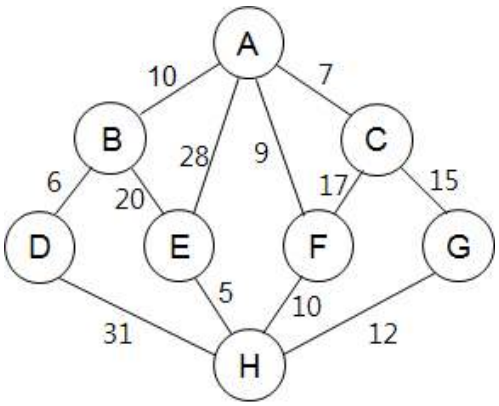
변수의 생성/소멸	LDR	출력	순서
p☆$_0$	~~LD~~R	A	5
p☆$_4$ p☆2	~~LDR~~ ~~LDR~~	B	2,7
p☆$_3$ p☆$_4$ p☆5 p☆6	~~LDR~~ ~~LDR LDR~~	D, E,G	1, 4, 8
p☆ p☆ p☆$_7$ p☆ p☆ p☆ p☆	~~LDR~~	H	3
p☆ p☆			

나. postOrder 운행의 과정 추적에서 다음 단계를 완성하시오.

*postOrder 운행(미완)

변수의 생성/소멸	LRD	출력	순서
p☆$_0$	~~LRD~~	A	8
p☆$_1$ p☆2	~~LRD LRD~~	B,C	4,7
p☆$_3$ p☆$_4$ p☆5 p☆6	~~LRD~~ ~~LRD~~ ~~LRD LRD~~	D,E,F,G	1,3,5,6
p☆ p☆ p☆$_7$ p☆	~~LRD~~	H	2
p☆ p☆			

19. 다음 그래프를 인접행렬로 표현하시오.



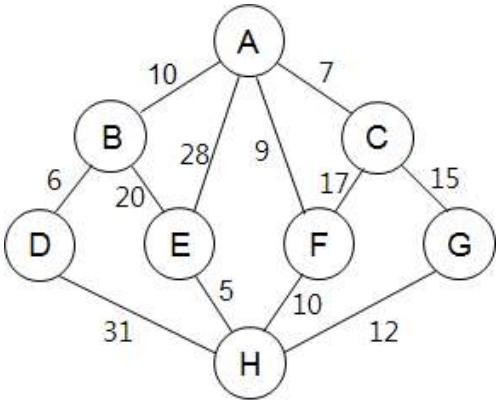
정점집합: V

A	B	C	D	E	F	G	H
0	1	2	3	4	5	6	7

0	1	1	0	1	1	0	0
1	0	0	1	1	0	0	0
1	0	0	0	0	1	1	0
0	1	0	0	0	0	0	1
1	1	0	0	0	0	0	1
1	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1
0	0	0	1	1	1	1	0

인접행렬

20. 최소비용 신장트리



가. Kruskal Algorithm으로 최소비용 신장트리를 단계별로 그리시오.

[1단계]
E-H

[2단계]
E-H B-D

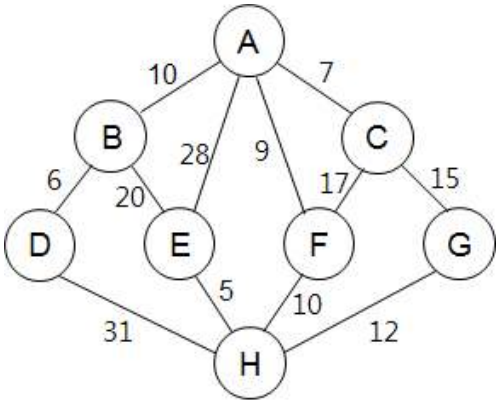
[3단계]
E-H B-D A-C

[4단계]
E-H B-D A-C
A-F

[5단계]
E-H B-D A-C
A-F A-B

[6단계]
E-H B-D A-C
A-F A-B F-H

[7단계]
E-H B-D A-C
A-F A-B F-H
H-G



나. Prim Algorithm으로 최소비용 신장트리를 단계별로 그리시오. 출발은 'H'

[1단계]
H-E

[2단계]
H-E H-F

[3단계]
H-E H-F A-F

[4단계]
H-E H-F A-F
A-C

[5단계]
H-E H-F A-F
A-C A-B

[6단계]
H-E H-F A-F
A-C A-B B-D

[7단계]
H-E H-F A-F
A-C A-B B-D H-G