자료구조론	국가 전산 7급	2021년 9월 11일

♠ 필기합격인원/합격선(52명/84점) - 선발예정인원 42명 ♠

1. 다음 탐색 알고리즘에 대한 설명으로 옳지 않은 것은? [2021년 국가 7급]

```
int a_search(int list[], int key, int n)
{
  int i;
  for(i=0; i<n; i++)
    if(list[i]==key) return i;
  return -1;
}
```

① 탐색에 성공하면 키 값의 인덱스를 반환한다.
② 최선의 경우, 시간복잡도는 빅오 표기법으로 O(1)이다.
③ 최악의 경우, 시간복잡도는 빅오 표기법으로 O(n)이다.
④ 평균적인 경우, 시간복잡도는 빅오 표기법으로 O((n+1)/2)이다.

♤ 선형검색

- 평균적인 경우, 시간복잡도는 빅오 표기법으로 O((n+1)/2)이다.(×)
  → 먼저, 빅오 표기법으로 O((n+1)/2)처럼 표기하지 않는다.
  → O((n+1)/2)는 빅오 표기법으로 O(n)처럼 표기해야 한다.

// 주어진 문제는 선형검색 알고리즘이다.

- 선형검색은 첫 번째 또는 마지막 레코드를 시작으로 탐색 작업을 순차적으로 처리한다.
- 선형검색은 프로그램 작성이 쉽다.
- 선형검색은 정렬되지 않은 레코드 검색이 가능하다.
- 선형검색은 파일이 크면 탐색시간이 증가한다.
- 선형검색에서 최선의 경우 : 탐색 대상이 첫 번째에 있을 경우 O(1)
- 선형검색에서 최악의 경우 : 탐색 대상이 마지막에 있을 경우 O(n)
- **선형검색의 평균검색장(L)** = $\frac{1+2+3+\cdots+n}{n} = \frac{\frac{n(n+1)}{2}}{n} = \frac{n+1}{2}$ = O(n)

정답 : ④

2. 다음 C 코드의 실행 결과는? [2021년 국가 7급]

```
#include <stdio.h>
int func (int n)
{
  if(n <= 1) return 2;
  return func (n-1) + n;
}
int main (int argc, char* argv[])
{
  printf("%d\n", func (10) );
  return 0;
}
```

① 55 ② 56
③ 58 ④ 60

♤ **재귀호출**

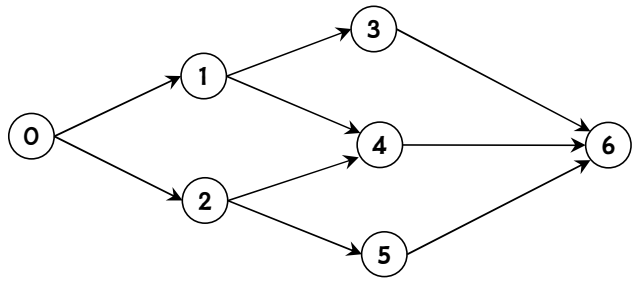
```
if(n <= 1) return 2;      // 완료조건, n=1일 때, 반환값이 2인 것에 주의!
return func (n-1) + n;
```

↓
↓ func(10) = ?
↓

- func(1) = 2
- func(2) = func(1) + 2 = 2 + 2 = 4
- func(3) = func(2) + 3 = 4 + 3 = 7
- func(4) = func(3) + 4 = 7 + 4 = 11
- func(5) = func(4) + 5 = 11 + 5 = 16
- func(6) = func(5) + 6 = 16 + 6 = 22
- func(7) = func(6) + 7 = 22 + 7 = 29
- func(8) = func(7) + 8 = 29 + 8 = 37
- func(9) = func(8) + 9 = 37 + 9 = 46
- func(10) = func(9) + 10 = 46 + 10 = **56**

정답 : ②

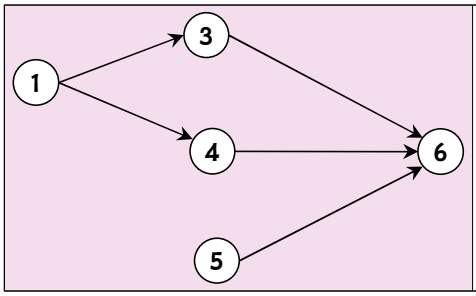
3. 다음 그래프에 대해 위상정렬(topological sort) 알고리즘을 수행했을 때 생성되는 위상순서 중 옳지 않은 것은? [2021년 국가 7급]



① 0, 1, 2, 3, 4, 5, 6
② 0, 1, 3, 2, 4, 5, 6
③ 0, 2, 4, 5, 1, 3, 6
④ 0, 2, 5, 1, 3, 4, 6

♤ **위상정렬**

// 다음은 주어진 그래프에서 0과 2를 선택한 경우이다.



③ 0, 2, 4, 5, 1, 3, 6
↓
4가 1보다 먼저 나타날 수 없다.

- 0과 2를 선택하였을 때는 1 또는 5를 선택할 수 있다.
- 1을 선택한 후에 4를 선택할 수 있다.

// 위상정렬(topological sort) - 정점 작업 네트워크

- 선후 관계가 존재하는 AOV 네트워크에서 만들어지는 순서리스트를 **위상순서**라 하고,
- 이런 정렬을 위상정렬(topological sort)이라 한다.
- 위상정렬 순서는 여러 가지가 있을 수 있다.
- 위상정렬은 **부분순서**(partial order)가 존재한다.
- 비사이클 방향그래프이어야 위상정렬이 가능하다.(방향사이클이 없어야 한다)

정답 : ③

4. 다음과 같이 배열을 이용하여 스택 자료구조를 정의하였다. ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은? [2021년 국가 7급]

```
-----------------------------------
#define MAX 10
int stack[MAX];
int top = -1;
int push(int t) {
  if (top >= MAX - 1) {
    printf("\n Stack overflow.");
    return -1;
  }
  stack[   ㉠   ] = t;
  return t;
}
int pop(void) {
  if (top < 0) {
    printf("\n Stack underflow.");
    return -1;
  }
  return stack[   ㉡   ];
}
-----------------------------------
```

	㉠	㉡
①	++top	--top
②	++top	top--
③	top++	--top
④	top++	top--

♤ **스택에서 입출력**

push	• top 포인터 증가 후, 자료입력 • stack[++top] = t;
pop	• 자료삭제 후, top 포인터 감소 • return stack[top--];

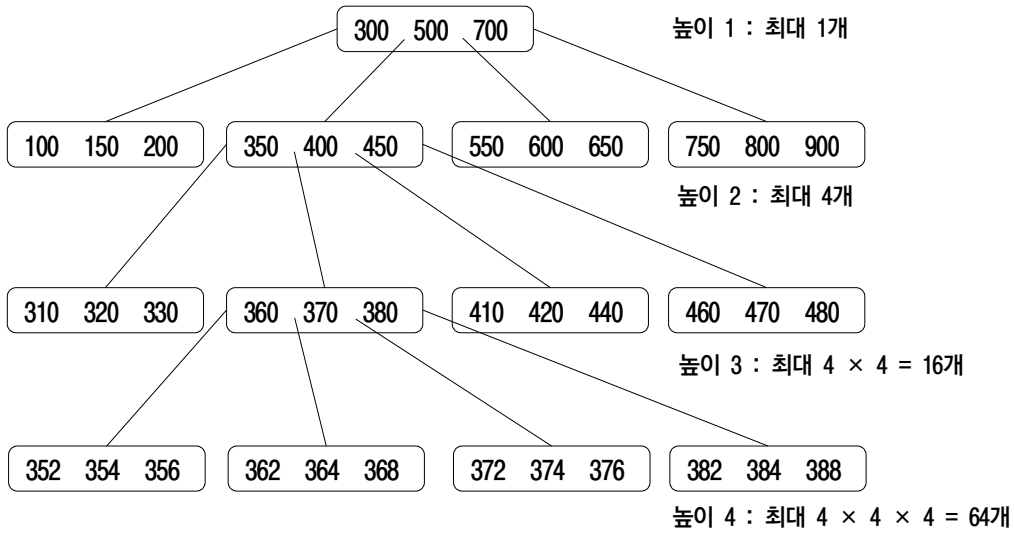
정답 : ②

5. 차수가 4이고 높이가 4인 m원 탐색트리가 가질 수 있는 노드의 최대 수는? (단, 루트노드만 있는 경우 트리의 높이는 1이다) [2021년 국가 7급]

① 21 ② 51 ③ 85 ④ 255

♤ **m원 탐색트리**

// 차수가 4이고 높이가 4인 m원 탐색트리를 개략적으로 그리면 다음과 같다.



• **최대 노드수** = 1 + 4 + 16 + 64 = **85**

// 공식으로 풀면

• m=4이므로, 각 노드는 최대 m − 1 = 4 − 1 = 3(개)의 키를 가질 수 있다.

• m원 탐색트리가 **최대 노드수**를 가지려면, **균형트리**구조이어야 한다.

• 차수가 m이고 높이가 h이면, **최대 노드수**는 $(m^h-1)/(m-1)$이다.

• 차수가 m이고 높이가 h이면, **최대 원소수**는 (m^h-1)이다.

• m=4이고, h=4이면, **최대 노드수** = $(m^h-1)/(m-1)$ = $(4^4-1)/(4-1)$ = **85**

// m원 탐색트리

• 이진탐색트리를 일반화시킨 형태로 모든 노드들이 m 이하의 차수를 갖는 탐색트리이다.

• 각 노드는 최대 m−1개의 키를 가질 수 있다.

• m원 탐색트리는 균형트리가 아니다. 한쪽 방향으로 기울어질 수 있다.

• m원 탐색트리가 **최대 노드수**를 가지려면, **균형트리**구조이어야 한다.

정답 : ③

6. 다음은 이진트리를 전위순회하여 얻어진 전위표기식이다. 이 트리에 대한 설명으로 옳지 않은 것은? [2021년 국가 7급]

```
---------------------------------
+ / * 6 2 - 3 1 * + 5 8 - 7 4
---------------------------------
```

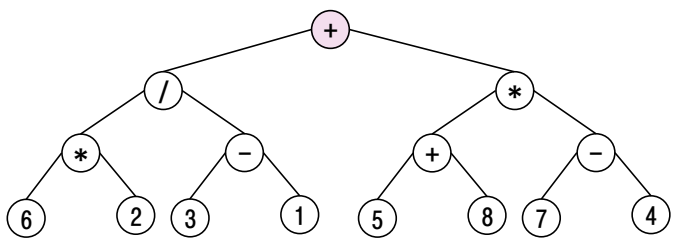
① 연산식의 최종 결과는 45이다.
② 트리로 재구성하면 루트노드는 +이다.
③ 트리로 재구성하여 중위순회하면 6*2/3-1+5+8*7-4이다.
④ 트리로 재구성하여 후위순회하면 62*31/-58+74-*+이다.

♤ **이진트리 순회**

// 전위표기식을 중위표기식으로 변경

+ / * 6 2 − 3 1 * + 5 8 − 7 4	← 전위표기식
↓	주어진 식을 **뒤에서 앞**으로 읽으면서 연산자를 만나면 ( )로 묶는다.
+ / * 6 2 − 3 1 * + 5 8 (7 − 4)	
+ / * 6 2 − 3 1 * (5 + 8) (7 − 4)	
+ / * 6 2 − 3 1 (5 + 8) * (7 − 4)	
+ / * 6 2 (3 − 1) (5 + 8) * (7 − 4)	
+ / (6 * 2) (3 − 1) (5 + 8) * (7 − 4)	
+ (6 * 2) / (3 − 1) (5 + 8) * (7 − 4)	
↓	
(6 * 2) / (3 − 1) + (5 + 8) * (7 − 4)	← 중위표기식 완성, 연산 결과는 45

↓
↓ **트리로 구성하면**
↓



루트노드는 +이다.

- **중위순회(좌,중간,우) :** 6 * 2 / 3 − 1 + 5 + 8 * 7 − 4
- **후위순회(좌,우,중간) :** 6 2 * 3 1 − / 5 8 + 7 4 − * +

정답 : ④

7. 이진트리의 성질에 대한 설명으로 옳은 것은? (단, 루트노드의 레벨은 0이고, 차수가 0인 노드의 개수는 n_0, 차수가 1인 노드의 개수는 n_1, 차수가 2인 노드의 개수는 n_2로 한다) [2021년 국가 7급]

① 이진트리에서 단말노드의 개수를 n_0이라 할 때, n_2와의 관계식 $n_0 = n_2 + 1$을 만족한다.

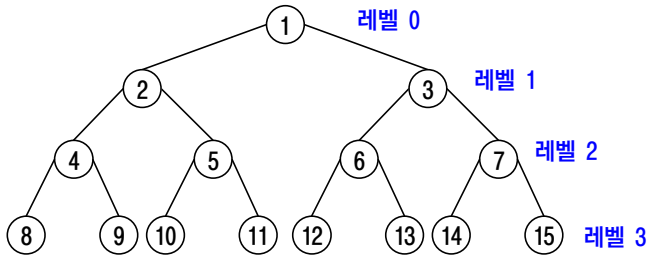
② 깊이가 k인 이진트리가 가질 수 있는 노드의 최대 개수는 $2^{(k+1)}$이다.(k≥0)

③ 이진트리의 레벨 j에 있는 노드의 최대 개수는 $2^{(j-1)}$개이다.(j≥0)

④ 이진트리의 전체 노드 개수는 $n_0 + n_1 + n_2 - 1$개이다.

♤ **이진트리 성질**

// 이런 유형의 문제는 다음처럼 트리를 그려서 풀면된다.



↓

↓ **루트노드의 레벨은 0인 경우**

↓

① n_0는 차수가 0, n_2는 차수가 2인 노드 수라 할 때 $n_0 = n_2 + 1$이다.

• $n_0 = 8$

• $n_2 = 7$이므로 $n_0 = n_2 + 1$

② 깊이가 k인 이진트리는 최대 $2^{(k+1)}-1$개의 노드를 가질 수 있다.

• k = 3일 때, 최대 노드수 = $2^{(k+1)}-1 = 2^4 - 1 =$ **15**(개)

③ 이진트리의 레벨 j에 있는 노드의 최대 개수는 2^j개다.(j≥0)

• j = 2일 때, 레벨 j에 있는 노드의 최대 개수는 $2^j = 2^2 =$ **4**개다.

④ 이진트리의 전체 노드 개수는 $n_0 + n_1 + n_2$개다.

• $n_0 + n_1 + n_2 = 8 + 0 + 7 =$ **15**(개)

[주의!] 주어진 문제에서는 **루트노드의 레벨을 0**으로 제시하였다.

정답 : ①

8. 다음 C 언어 코드를 실행한 후 5번째 및 7번째 줄에 출력되는 문장으로 옳은 것은? [2021년 국가 7급]

```
#include <stdio.h>
void hanoi(int n, char from, char tmp, char to) {
  if( n==1 )
    printf("disk 1 from %c to %c\n", from, to);
  else {
    hanoi(n-1, from, to, tmp);
    printf("disk %d from %c to %c\n", n, from, to);
    hanoi(n-1, tmp, from, to);
  }
}
int main(int argc, char* argv[]) { hanoi(3, 'A', 'B', 'C'); }
```

	5번째 줄	7번째 줄
①	disk 1 from B to A	disk 1 from A to C
②	disk 1 from C to B	disk 2 from B to C
③	disk 2 from A to B	disk 3 from A to C
④	disk 3 from A to C	disk 2 from B to C

♤ **하노이타워(Tower of Hanoi) - 재귀호출**

• hanoi(3, 'A', 'B', 'C'); → 원판은 3개이고, 'A', 'B', 'C'는 3개의 기둥을 의미한다.

// 실행 결과는 다음과 같다.

1번째 줄 : disk 1 from A to C → 가장 작은 원판인 disk 1이 기둥 A에서 기둥 C로 이동
2번째 줄 : disk 2 from A to B
3번째 줄 : disk 1 from C to B
4번째 줄 : disk 3 from A to C → **가장 큰 원판**인 disk 3이 **중간**에 이동된다.
5번째 줄 : disk 1 from B to A
6번째 줄 : disk 2 from B to C
7번째 줄 : disk 1 from A to C

• **하노이타워** 문제는 **가장 큰 원판**이 이동되는 시점이 전체 이동횟수에서 **항상 중간**이 된다.

정답 : ①

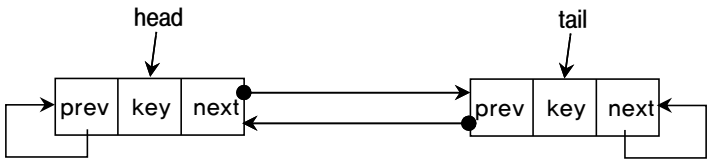
9. 다음과 같이 이중연결리스트를 이용하여 큐 자료구조를 정의하였다. add는 큐의 삽입 알고리즘이고, delete는 큐의 삭제 알고리즘이다. ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은? [2021년 국가 7급]

```
typedef struct _dnode {
  int key;
  struct _dnode *prev, *next;
} dnode;
dnode *head = (dnode*)malloc(sizeof(dnode));
dnode *tail = (dnode*)malloc(sizeof(dnode));
head->prev = head; head->next = tail;
tail->prev = head; tail->next = tail;
int add(int k)
{
  dnode *t = (dnode*)malloc(sizeof(dnode));
  if (t == NULL) {
    printf("\n Out of memory.");
    return -1;
  }
  t->key = k;
              ㉠
  tail->prev = t; t->next = tail;
  return k;
}
int delete(void)
{
  dnode *t; int i;
  t = head->next;
  if (t == tail) {
    printf("\n Queue underflow.");
    return -1;
  }
  i = t->key;
              ㉡
  free(t); return i;
}
```

① ㉠ tail->prev = t; t->prev = tail->prev;
㉡ head->next = t->next; t->next->prev = head;
② ㉠ tail->prev->next = t; t->prev = tail->prev->next;
㉡ head->next = t->next; t->next->prev = head;
③ ㉠ tail->prev->next = t; t->prev = tail->prev;
㉡ head->next = t->next; t->next = head;
④ ㉠ tail->prev->next = t; t->prev = tail->prev;
㉡ head->next = t->next; t->next->prev = head;

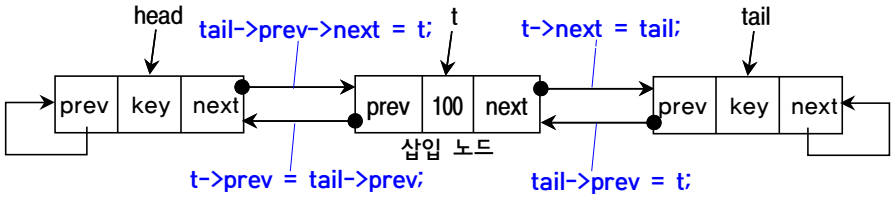
♤ 이중연결리스트를 이용한 자료구조 큐

// 초기상태



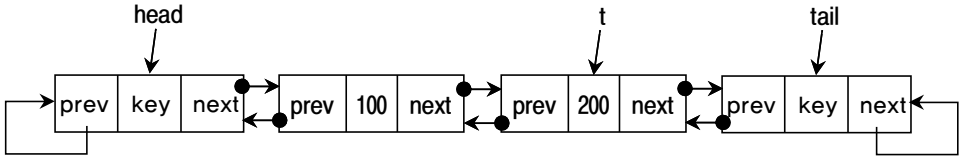
head->prev = head;
head->next = tail;
tail->prev = head;
tail->next = tail;

// 큐에 키 100을 삽입 : add(100)



㉠ tail->prev->next = t; t->prev = tail->prev;

// 큐에 키 200을 삽입 : add(200)



// 큐에서 삭제 : delete(void)

• 큐에서 삭제 : head->next가 가리키는 노드를 삭제한다.

↓t = head->next;

㉡ head->next = t->next; t->next->prev = head;

정답 : ④

10. 다항식에서 하나의 항을 표현하기 위해 다음과 같은 구조체 poly를 정의하였다. 이 구조체를 이용하여 $A(x)=3x^{14}+2x^{8}+1$과 $B(x)=8x^{14}-4x^{9}+10x^{3}$을 저장하는 데 필요한 최소의 메모리 용량(byte)은? (단, 정수(int)를 표현하는 데 2byte를 사용하고, 실수(float)를 표현하는 데 4byte를 사용하며, 다항식의 모든 항은 poly 구조체를 이용하여 저장한다) [2021년 국가 7급]

```
---------------
 struct poly
 {
   int exp;
   float coef;
 };
---------------
```

① 22 ② 24 ③ 36 ④ 38

♤ **다항식(polynomial) 배열 표현 – 최소 메모리 용량**

표현	• **항수**와 **계수가 0이 아닌 항**만을 **(지수, 계수)** 쌍으로 표현한다. • 표현 : (**항수**, **지수, 계수, 지수, 계수**, …, **지수, 계수, 지수, 계수**)

• 표현에서 '**항수**'는 반드시 필요한 것은 아니다.

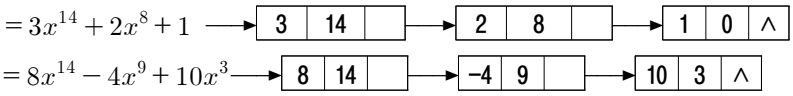
• **지수**(int exp;)는 정수이므로 **2**byte가 필요하다.
• **계수**(float coef;)는 실수이므로 **4**byte가 필요하다.

// **계수가 0이 아닌 항**만을 **(지수, 계수)** 쌍으로 표현한다.

다항식	배열 표현	최소 메모리 용량
$A(x)=3x^{14}+2x^{8}+1$	(14, 3, 8, 2, 0, 1)	(2 + 4) × 3 = **18**
$B(x)=8x^{14}-4x^{9}+10x^{3}$	(14, 8, 9, −4, 3, 10)	(2 + 4) × 3 = **18**
∴ 전체 용량 = 18 + 18 = **36**		

◈ **참고 – 다항식의 연결리스트 표현**

다항식의 연결리스트 표현은 | 계수 | 지수 | 링크 | 라는 3개의 필드를 이용한다.



정답 : ③