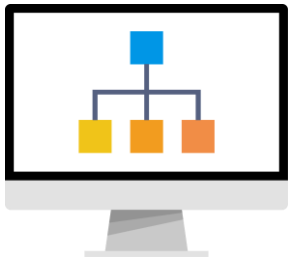


Data Structure & Algorithm
자료구조 및 알고리즘

15. 코드 리뷰 및 프로젝트 설명

강의 내용

- 문제 풀이 (13, 14강 온라인 보고서, 재귀 호출)
- 과제 1 설명
- 간단한 코드 리뷰

13강 보고서 돌아보기

- 노드가 n개인 트리의 간선 수의 최솟값/최댓값은? **n-1**

- 높이가 n인 포화 이진 트리의 개수는? **1**

- 높이가 n인 포화 이진 트리의 노드 수는? $\mathbf{2^{n+1} - 1}$

13강 보고서 돌아보기

- 높이가 n인 완전 이진 트리의 개수는? $\boldsymbol{2^n}$

- 높이가 n인 완전 이진 트리의 내부 노드 수는? $\boldsymbol{2^n - 1}$

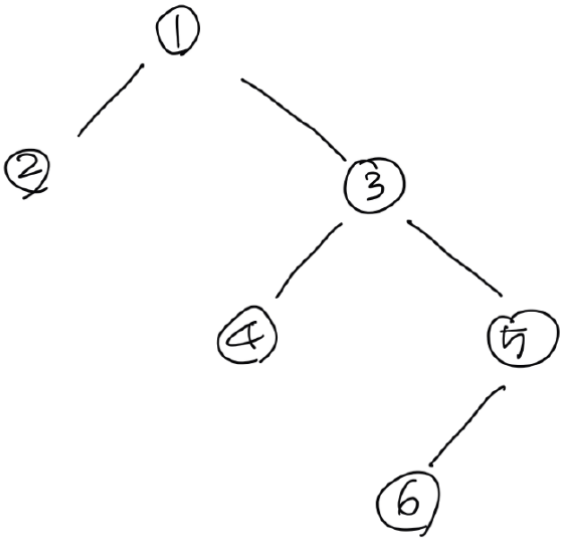
- 높이가 n인 정 이진 트리의 노드의 수의 최솟값/최댓값은?
  - 최솟값: $\boldsymbol{2n + 1}$
  - 최댓값: $\boldsymbol{2^{n+1} - 1}$

14강 보고서 돌아보기

- 어떤 트리의 각 노드를 색으로 칠하고 싶다. 단, 간선으로 연결된 두 노드는 다른 색으로 칠해야 한다. 몇 가지 색이 필요할까?
  - 2가지
  - 짝수 레벨은 빨간색으로, 홀수 레벨은 파란색으로

14강 보고서 돌아보기

- 어떤 트리의 전위 순회 결과가 “1 2 3 4 5 6”이고 중위 순회 결과가 “2 1 4 3 6 5”라고 한다. 이 트리는 어떻게 생겼을까?
  - (전위, 중위) 또는 (후위, 중위) 순회 결과를 알면 트리를 다시 만들 수 있다.
  - (전위, 후위) 는 왜 안될까?



14강 보고서 돌아보기

- 어떤 트리에 대해 함수 *f*는 노드의 번호 *i*를 인자로 받아 여기서 가장 멀리 떨어진 노드 번호 *j*를 돌려준다고 하자. *f*를 최소 몇 번 호출해야 트리에서 가장 멀리 떨어진 두 노드를 찾을 수 있을까?
  - 2번
  - 트리에서 아무 노드 i를 잡고 f(i)를 호출하여 가장 멀리 떨어진 u를 찾는다.
  - f(u)를 호출하여 u에서 가장 멀리 떨어진 노드 v를 찾는다. (u, v)가 정답.
  - (u, v)의 거리를 트리의 **지름**이라고 부른다.

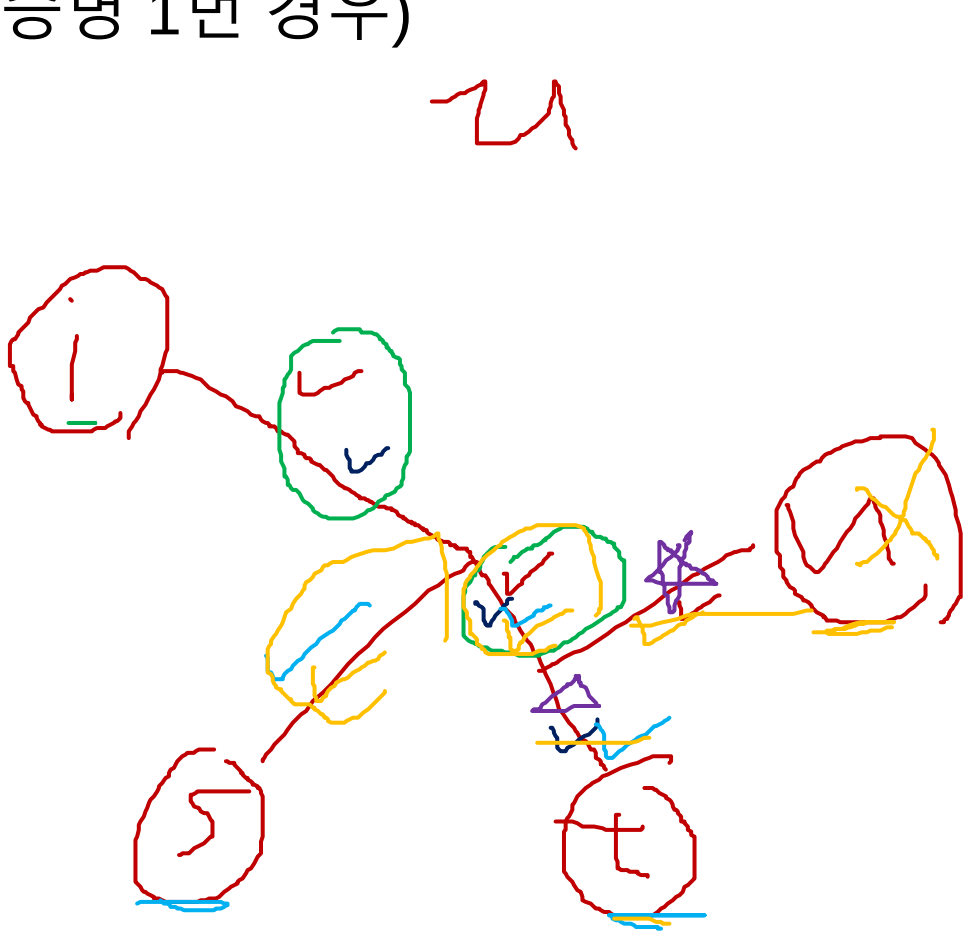
14강 보고서 돌아보기

- i: 임의로 선택한 노드
- u: i에서 가장 멀리 떨어진 노드 f(i)
- s, t: 실제로 가장 멀리 떨어져있는 노드

- u가 s또는 t임을 보이자.
  - u가 s도 아니고 t도 아니라고 가정하면 모순이 발생함을 보이자 (귀류법).

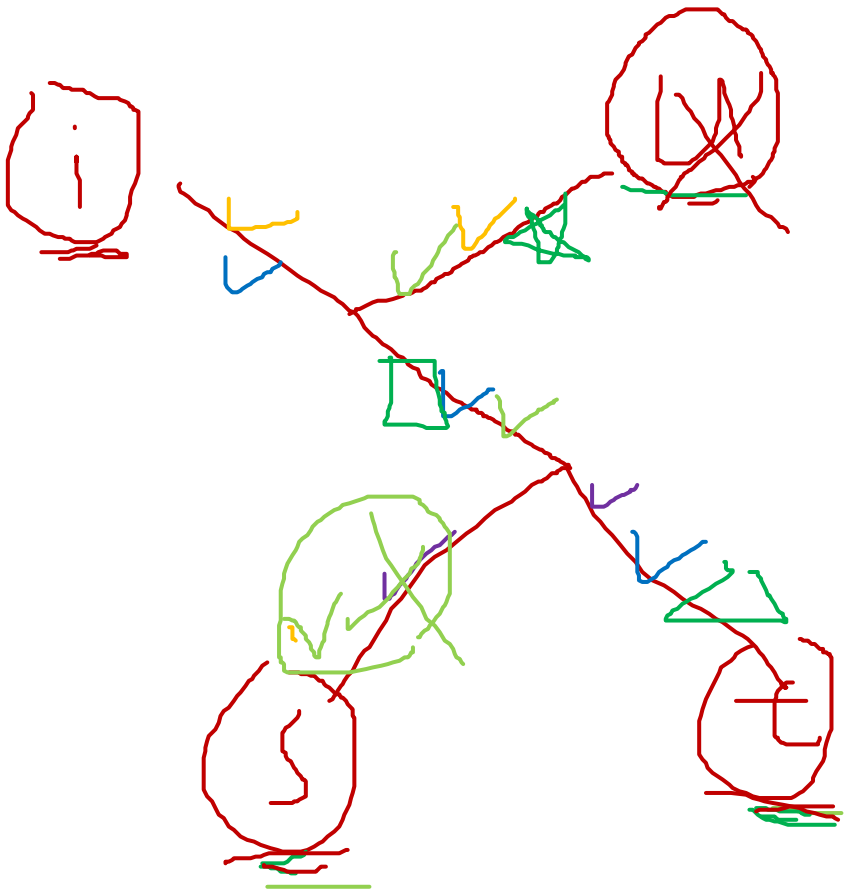
14강 보고서 돌아보기

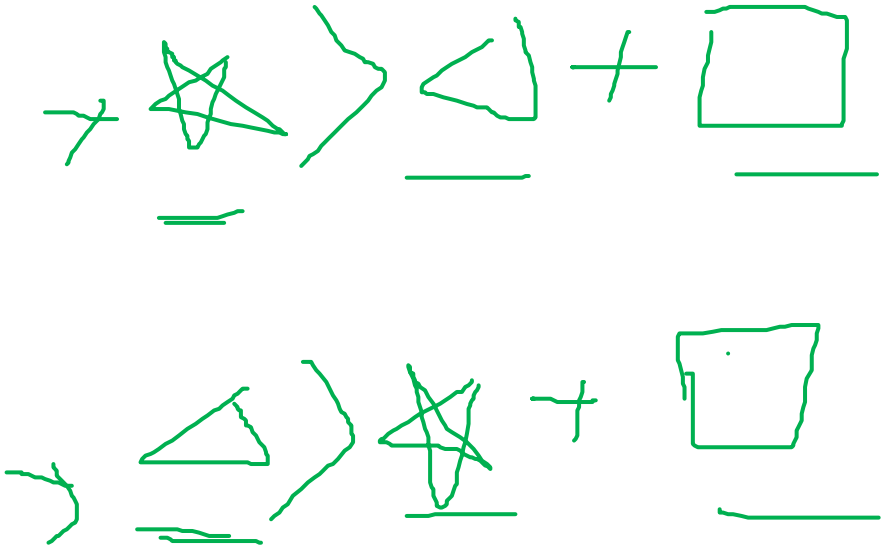
- 증명 1번 경우)



14강 보고서 돌아보기

- 증명 2번 경우)





재귀 호출

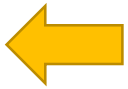
```
1  #include <stdio.h>
2
3  int n;
4  char str[20];
5
6  void fill(int i) {
7      if(i == n) {
8          str[i] = '\0';
9          printf("%s\n", str);
10         return;
11     }
12     str[i] = 'a';
13     fill(i + 1);
14
15     str[i] = 'b';
16     fill(i + 1);
17 }
18
19 int main() {
20     scanf("%d", &n);
21     fill(0);
22     return 0;
23 }
```

- str[i]에 ‘a’를 넣고 fill(i+1) 호출 (빨간 원으로 표시)
- str[i]에 ‘b’를 넣고 fill(i+1) 호출 (파란 원으로 표시)
- 원 안에 i를 표현
- 메인 함수에서 fill(0)을 호출했으므로 초기 i의 값은 0이다.

재귀 호출

```c
#include <stdio.h>

int n;
char str[20];

void fill(int i) {
    if(i == n) {
        str[i] = '\0';
        printf("%s\n", str);
        return;
    }
    str[i] = 'a';
    fill(i + 1);

    str[i] = 'b';
    fill(i + 1);
}

int main() {
    scanf("%d", &n);
    fill(0);
    return 0;
}
```

- i = 0
  - str: “a” 1
  - str: “b” 1



실행 위치

재귀 호출

```c
#include <stdio.h>

int n;
char str[20];

void fill(int i) {
    if(i == n) {
        str[i] = '\0';
        printf("%s\n", str);
        return;
    }
    str[i] = 'a';
    fill(i + 1);

    str[i] = 'b';
    fill(i + 1);
}

int main() {
    scanf("%d", &n);
    fill(0);
    return 0;
}
```

- i = 0
  - str: “a” 1
  - i=1
    - str: “aa” 2
    - str: “ab” 2
  - str: “b” 1

재귀 호출

```c
#include <stdio.h>

int n;
char str[20];

void fill(int i) {
    if(i == n) {
        str[i] = '\0';
        printf("%s\n", str);
        return;
    }
    str[i] = 'a';
    fill(i + 1);

    str[i] = 'b';
    fill(i + 1);
}

int main() {
    scanf("%d", &n);
    fill(0);
    return 0;
}
```

- i = 0
  - str: “a” 1
  - i=1
    - str: “aa” 2
      - i=2
      - str: “aaa” 3
      - str: “aab” 3
    - str: “ab” 2
  - str: “b” 1

재귀 호출

```c
#include <stdio.h>

int n;
char str[20];

void fill(int i) {
    if(i == n) {
        str[i] = '\0';
        printf("%s\n", str);
        return;
    }
    str[i] = 'a';
    fill(i + 1);

    str[i] = 'b';
    fill(i + 1);
}

int main() {
    scanf("%d", &n);
    fill(0);
    return 0;
}
```

- i = 0
  - str: “a” 1
  - i=1
    - str: “aa” 2
      - i=2
      - str: “aaa” 3
        - “aaa”출력 후 리턴
      - str: “aab” 3
    - str: “ab” 2
  - str: “b” 1

재귀 호출

```c
#include <stdio.h>

int n;
char str[20];

void fill(int i) {
    if(i == n) {
        str[i] = '\0';
        printf("%s\n", str);
        return;
    }
    str[i] = 'a';
    fill(i + 1);

    str[i] = 'b';
    fill(i + 1);
}

int main() {
    scanf("%d", &n);
    fill(0);
    return 0;
}
```

- i = 0
  - str: “a” 1
  - i=1
    - str: “aa” 2
      - i=2
      - str: “aaa” 3
        - “aaa”출력 후 리턴
      - str: “aab” 3
    - str: “ab” 2
  - str: “b” 1

재귀 호출

```c
#include <stdio.h>

int n;
char str[20];

void fill(int i) {
    if(i == n) {
        str[i] = '\0';
        printf("%s\n", str);
        return;
    }
    str[i] = 'a';
    fill(i + 1);

    str[i] = 'b';
    fill(i + 1);
}

int main() {
    scanf("%d", &n);
    fill(0);
    return 0;
}
```

- i = 0
  - str: “a” 1
  - i=1
    - str: “aa” 2
      - i=2
      - str: “aaa” 3
        - “aaa”출력 후 리턴
      - str: “aab” 3
        - “aab”출력 후 리턴
    - str: “ab” 2
  - str: “b” 1

과제 1 설명

목표

사용자로부터 명령어를 입력받아 100 개의 리스트에 대한 데이터 삽입/삭제/탐색/출력을 수행하는 프로그램을 작성한다. 주어진 뼈대 코드(main.h 와 main.c)를 활용하되, main.c 에 비어있는 함수를 채워넣는 식으로 구현한다.

주의사항

- 모든 명령어는 알파벳 소문자로 주어진다.
- 프로그램 내부에는 최대 100 개의 리스트가 있을 수 있다. 따라서, 아래 명세에서 〈id〉 값은 0 이상 99 이하의 정수이다.
- 리스트는 4 바이트 정수형 데이터를 저장한다. 따라서, 아래 명세에서 〈data〉 값은 항상 4 바이트 정수형 자료형으로 입력 및 표현할 수 있다.
- 예외적으로 출력 명세에 언급한 경우가 아니라면 항상 유효한 입력만 주어진다고 가정해도 좋다.
- 아래 예제 명세에서 😊 기호의 왼편은 예제 입력, 오른편은 예제 입력에 대한 예제 출력을 나타낸다.
- 한 명령어에 대한 출력이 끝나면 줄 바꿈 문자를 출력하여 다음 명령어의 입력이 콘솔의 맨 왼쪽 끝에서 이루어질 수 있도록 하라.

과제 1 설명

명세

프로그램은 표준 입력(scanf 를 이용)을 통하여 사용자에게 명령어를 입력 받고, 결과를 표준 출력(printf 를 이용)으로 출력한다. 종료 명령어가 입력될 때까지 반복적으로 명령어를 입력받아 처리해야 한다.
프로그램에서 지원하는 명령어는 아래와 같다.

insert ⟨id⟩ ⟨pos⟩ ⟨data⟩

동작 ⟨id⟩번째 리스트의 ⟨pos⟩ 위치에 ⟨data⟩를 삽입한다. ⟨id⟩번째 리스트의 길이가 n 일 때 ⟨pos⟩는 0 이상 n 이하의 정수이며 0 일 경우 데이터를 리스트의 머리 앞에 (새로운 머리가 된다), n 일 경우 데이터를 리스트의 꼬리 뒤에(새로운 꼬리가 된다) 삽입한다.

예외적으로 ⟨pos⟩의 값이 -1 일 때, 리스트의 꼬리 뒤에 삽입하는 것으로 간주한다.

만약, ⟨id⟩번째 리스트가 비어있을 경우 유효한 ⟨pos⟩의 값은 0 또는 -1 뿐이다. 두 경우 모두 ⟨data⟩는 리스트의 머리이자 꼬리가 된다.

출력 ⟨pos⟩의 값이 ⟨id⟩번째 리스트의 길이보다 크면 삽입에 실패하여 -1 을 출력하고, 삽입에 성공하면 1 을 출력한다.

예제 insert 0 0 3 😊 1

과제 1 설명

find ⟨id⟩ ⟨data⟩

동작 ⟨id⟩번째 리스트에서 ⟨data⟩를 찾아 그 인덱스를 출력한다. 만약 ⟨data⟩가 여러 번 등장한다면 제일 작은 인덱스를 한 번만 출력하면 된다. 출력되는 인덱스는 0 부터 시작한다. 따라서, 리스트의 길이가 n 일 때 머리에서 데이터를 찾으면 0 을, 꼬리에서 데이터를 찾으면 n-1 을 출력해야 한다. 찾지 못할 경우 -1 을 출력한다.

출력 ⟨id⟩번째 리스트에서 처음 등장하는 ⟨data⟩의 인덱스를 출력하고(0 부터 시작), 만약 찾지 못했다면 -1 을 출력한다.

예제 find 0 5 😊 1

find 0 4 😊 -1 (리스트 0 에는 위의 insert 예제에서 처럼 3 과 5 가 있으므로 4 는 존재하지 않음)

과제 1 설명

delete ⟨id⟩ ⟨pos⟩

동작 ⟨id⟩번째 리스트에서 ⟨pos⟩번째 데이터를 삭제한다. ⟨id⟩번째 리스트의 길이가 n 일 때 ⟨pos⟩는 0 이상 n 미만의 정수이며 0 일 경우 리스트의 머리를, n-1 일 경우 리스트의 꼬리를 삭제한다.

예외적으로 ⟨pos⟩의 값이 -1 일 때, 리스트의 꼬리를 삭제하는 것으로 간주한다.

출력 ⟨pos⟩의 값이 ⟨id⟩번째 리스트의 길이보다 크거나 같으면 삭제에 실패하여 -1 을 출력하고 삭제에 성공하면 1 을 출력한다.

예제 delete 0 0 😊 1 (리스트 0 에는 현재 3 5 가 있는데 이 delete 명령어를 수행하면 5 만 남게 된다)

delete 1 0 😊 -1 (리스트 1 은 비어있으므로 0 번째 요소를 삭제할 수 없다)

과제 1 설명

count ⟨id⟩

동작 ⟨id⟩번째 리스트의 길이를 출력한다.

출력 ⟨id⟩번째 리스트의 길이를 출력한다. 비어있다면 0 을 출력한다.

예제 count 0 😊 1

reset ⟨id⟩

동작 ⟨id⟩번째 리스트의 데이터를 모두 삭제하고 초기화한다.

출력 이 명령어의 출력은 없다.

예제 reset 0

과제 1 설명

print ⟨id⟩

동작 ⟨id⟩번째 리스트에 저장된 데이터를 앞에서부터 하나씩 출력한다.

출력 ⟨id⟩번째 리스트에 저장된 데이터를 앞에서부터 하나씩 띄어쓰기로 구분하여 출력한다. 구현의 편의를 위해 마지막 데이터를 출력하고 그 다음에 띄어쓰기를 출력하는 것을 허용한다. 만약 리스트가 비어있다면 줄바꿈 문자 하나만 출력하라.

예제 (0 번째 리스트에 3, 4, 5 가 저장되어 있다고 가정)

print 0 😊 3 4 5

print_reverse ⟨id⟩

동작 ⟨id⟩번째 리스트에 저장된 데이터를 뒤에서부터 하나씩 출력한다.

출력 ⟨id⟩번째 리스트에 저장된 데이터를 뒤에서부터 하나씩 띄어쓰기로 구분하여 출력한다. 구현의 편의를 위해 마지막 데이터를 출력하고 그 다음에 띄어쓰기를 출력하는 것을 허용한다. 만약 리스트가 비어있다면 줄바꿈 문자 하나만 출력하라.

예제 (0 번째 리스트에 3, 4, 5 가 저장되어 있다고 가정)

print_reverse 0 😊 5 4 3

과제 1 설명

제출

- 기한: 2020 년 5 월 10 일 일요일 23:59
- 방법: 포털의 과제 제출란에 정해진 이름으로 압축 파일을 올린다. 학번과 이름이 20171001 김덕성이라면 HW1_20171010_김덕성.zip 으로 아래 파일을 압축하여 제출한다.
- **main.c**: 위의 명세를 구현한 소스 코드 파일
- **report.pdf**: 구현 방법을 요약한 보고서. 단, 보고서는 A4 용지로 2 장 이내로 제한한다.
- 딜레이는 전체 점수에서 1 일 이내(5 월 11 일 23:59)인 경우 30%, 3 일 이내(5 월 13 일 23:59)인 경우 50%, 7 일 이내(5 월 17 일 23:59)인 경우 70%를 감점한다.

과제 1 설명

- 헤더 파일을 수정해도 되나요?
  - 안됩니다.

- 코드에 이미 return 0으로 되어있는데 이 부분을 수정해도 되나요?
  - 됩니다.

코드 리뷰

- 수강생들의 실제 사례를 가지고 어떻게 개선할 수 있을지 고민해 봅시다.

- 협업을 할 때 서로의 발전을 위해서 필수적인 과정!

사례1: 불필요한 else if, else

```
int fibo(int n) {
    if (n == 1) return 1;
    else if (n == 2) return 1;
    else if (n == 3) return 2;
    else return fibo(n-1) + fibo(n - 2) + fibo(n - 3);
}

int fibo(int n) {
    if (n == 1) return 1;
    if (n == 2) return 1;
    if (n == 3) return 2;
    return fibo(n-1) + fibo(n - 2) + fibo(n - 3);
}
```

사례2: 거듭제곱 구하기

```
for (i = 0; i < 2 ^ n; i++) {

}

// ----------------------------

int pow2(int n) {
    int res = 1;
    while(n--) res *= 2;
    return res;
}

// pow(n)
```

사례3: 불필요한 초기화

```
int k = 0;
int count = 0;
int cnt[16] = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 };

// ---------------------------------------

int k, count, cnt[16];
```

사례4: 예약어로 변수/함수 정의

```
void bool(char x[], int i, int n)
{
    int j = 0;
    int k = 0;
}

// ----------------------------------------

void recur(char x[], int i, int n)
{
    int j = 0;
    int k = 0;
}
```

사례5: free 잊어버림

```
int main(void) {
    // ...
    b = (int*)malloc(sizeof(int) * cal(a));
    full(0,a,b);
    return 0;
}
```

사례6: 불필요한 strcpy나 strcat

```
strcpy(&str[i-n], "a");
printAB(n-1, i);
strcpy(&str[i-n], "b");
printAB(n-1, i);

// ----------------------

str[i-n]='a';
printAB(n-1, i);
str[i-n]='b';
printAB(n-1, i);
```

출석 인정을 위한 보고서 작성

- 포화 k진 트리(포화 3진 트리, 4진 트리..)가 있다고 가정하자. 아래 문제에 대한 답을 n과 k에 대한 식으로 나타내어라.

- 레벨 n에는 몇 개의 노드가 있을까?

- 높이 n인 포화 k진 트리에는 몇 개의 노드가 있을까?