설계

차리서 <reeseo@konkuk.ac.kr>

건국대학교 공과대학 컴퓨터공학부

학기 전체 일정

주	월	화	수	목	금	토	일	실습	강의
1	8月 28	29	30	31	1	2	3	수강 정정, 팀 결성	과목 오리엔테이션, 팀 결성 안내
2	9月 4	5	6	7	8	9	10	팀 결성 완료, 주제 선정 시작	주제 선정 안내, 기획서 안내
3	11	12	13	14	15	16	17	주제 선정 완료, 기획 시작	기획서 안내
4	18	19	20	21	22	23	24	**기획서 작성**	**설계 문서 안내**
5	25	26	27	28	29	30	1	설계 문서 작성	(설계 문서 문답)
6	10月 2	3	4	5	6	7	8	설계 문서 작성	요구사항 분석 안내
7	9	10	11	12	13	14	15	요구사항 분석 및 재설계	(요구사항 분석, 재설계 문답)
8	16	17	18	19	20	21	22	요구사항 분석 및 재설계	구현 및 검사 안내, 중간 발표 안내
9	23	24	25	26	27	28	29	구현 및 검사	(구현 및 검사 문답)
10	30	31	1	2	3	4	5	구현 및 검사	(구현 및 검사 문답)
11	11月 6	7	8	9	10	11	12	중간 발표	—
12	13	14	15	16	17	18	19	요구사항 분석, 재설계/구현	(재설계/구현 문답)
13	20	21	22	23	24	25	26	요구사항 분석, 재설계/구현	(재설계/구현 문답)
14	27	28	29	30	1	2	3	요구사항 분석, 재설계/구현, 검사	(재설계/구현 문답)
15	12月 4	5	6	7	8	9	10	요구사항 분석, 재설계/구현, 검사	기말 발표 안내
16	11	12	13	14	15	16	17	기말 발표	—

(팀 단위) 제출 일정

9/27(수) 오전 11:00 까지 (주의: 강의자료 #1, #2의 달력 표시(붉은 선)보다 하루 당겨짐)

- 초기 기획서 원판

10/9(월) 오전 11:00 까지 (주의: 강의자료 #1, #2의 달력 표시(붉은 선)보다 하루 당겨짐)

- 초기 기획서 수정판 (수정할 필요가 있는 팀만 제출하면 됨)
- 초기 설계 문서

10/23(월) 오전 11:00 까지 (단, 이 시한은 2 ~ 3 일 늦춰질 수도 있음: 추후 재공지)

- 1차 요구사항 분석서
- 1차 재설계 문서

11/6(월) 오전 11:00 까지

- 1차 요구사항 분석서 수정판 (수정할 필요가 있는 팀만 제출하면 됨)
- 1차 재설계 문서 수정판 (수정할 필요가 있는 팀만 제출하면 됨)
- 1차 구현물
- 1차 검사 보고서

기획 vs. 설계: 무엇을 적는 문서인가?

기획서

- 기획자/요구자가 정확히 **무엇**을 의도/요구했는가?
- 사용자 관점에서, 무슨 일을 하는 프로그램인가?
- 어떤 입력에 어떤 출력? 입력 방식은? 출력 방식은?
- 어떤 입력을 오류로 간주하고 뱉어내는가? 무엇을 할 수 있고, 할 수 없는가?

설계 문서

- 기획대로 동작하게 만들려면 **어떻게** 만들어야 하는가?
- 어떤 자료형의 어떤 변수들을 준비해야 하는가?
- 무엇을 리터럴로 하드 코딩하고, 무엇을 변수나 상수로 정의해 두어야 하는가?
- 어떤 기능을 따로 함수나 서브루틴으로 빼 두어야 하는가? 함수의 인자는?
- 어떤 클래스들과 상속 관계가 필요한가? 어떤 인스턴스들이 필요한가?
- 분기 지점과 조건은 무엇이고, 루프는 어디서 얼마나 돌아야하는가? (순서도)

기획 vs. 설계: 누구 읽으라는 문서인가?

기획서 (요구분석가/기획자가 작성)

- (다음 단계인) 설계자: 기획 내용을 보고, 그대로 설계함.
- (반복 단계인) 검사자: 자체 모순이 없는지 확인
- (이전 단계인) 요구자: 기획 내용을 보고, 자신의 요구대로 쓰여있는지 확인함.
  - 별도의 요구자가 존재하는 경우, 기획서(요구사항 분석서)는 **요구자도 알아볼 수 있어야** 한다!

설계 문서

- (다음 단계인) 구현자: 설계 내용을 보고, 그대로 구현함.
  - 구현자가 기획서를 직접 보고 곧바로 코드로 옮기는 것은 (비록 현실이더라도) 바람직하지 않음.
- (반복 단계인) 검사자: 설계가 기획에 부합하는지, 자체 모순이 없는지 확인
- 요구자는 설계 문서를 **못 알아보는 것이 정상**!
  - “못 알아봐야만 한다”가 아니라 “못 알아봐도 된다”임

설계 문서의 품질 (좋은 설계 문서의 요건)

- 기획에 부합하게 설계된 문서
- 누가 읽고 구현해도, 기능feature 은 물론 성능performance 까지 같은 구현물이 나오는 문서[1)]
- 구현자가 '이 방식으로 구현할까, 저 방식으로 구현할까?'라는 번민이 없는/적은 문서
- 미리 제한되어있는 구현 조건(플랫폼, 프로그래밍 언어 등)을 충분히 고려한 문서
- **추후 요구사항 변경에 따라 설계 변경도 용이하도록 미리 충분히 대비된 문서**
- [참고] 기계(프로그램)에 집어넣으면 자동으로 구현물(코드)이 나오는 문서
  - ('모든'은 아니지만) '일부' 설계를 구현물로 바꾸는 구현기는 실제로 존재함. (연구 진행 분야)
  - 설계 문서 자체가 정형적인formal 언어(논리식, 타입 표현식 등)로 작성되어야 가능함
  - 어떤 명제에 대한 (직관주의적) 증명은, 그 명제가 명세하는 대로 동작하는 (함수형) 프로그램임

"Proofs as Programs"
"Propositions as Specifications"

1) 기능은 '할 수 있는가 없는가', 성능은 '얼마나 (빠르게, 메모리를 덜 쓰면서) 잘 하는가'

설계 문서의 형식

그런 거 없음

- 구현 언어에 따라 설계 스타일이 **천차만별**
- 도구(문서 형식)에 따라 그에 맞는 방식으로만 생각하지 말 것
- 망치에 익숙해지고 망치질의 달인이 되면, 자칫 “바닥에 떨어뜨려 못을 박는” 발상을 못하게 됨
- 나사 박아야하는데 망치 들고 사용법 고민하지 말 것. 망치 쓸 장면이 아님.
- 대다수 학생들에게 가장 **자유도가 높고 제한이 적은 도구: 자연어, 표, 목록, 그림, 수식/논리식 등**

그런 거 있음

- 소프트웨어 공학 혹은 정형기법에서 학습
- 그 도구(설계 문서 형식)의 의의와 장점을 제대로 이해하기
- 그 도구(설계 문서 형식)의 한계와 단점을 놓치지 말고 비판적으로 받아들이기

설계 문서의 내용

기획 (요구사항) 대로 동작하려면 어떻게 해야 하는가? (구현 작전)

- 어떤 자료형, 자료 구조, 클래스, 클래스 멤버, 인스턴스, 인스턴스 멤버, 상속 관계들이 필요한가?
- 어떤 서브 루틴, 프로시저, 함수들이 필요한가?
- 무엇이 전역 변수고 무엇이 지역 변수인가? 무엇이 지역 함수인가?
- 어떤 변수가 불변 자료고 어떤 변수가 가변 자료인가?
- 기존 객체(의 멤버)에 새 값을 덮어쓸 것인가, 새 값을 (멤버로) 갖는 새 객체를 찍어낼 것인가?
- 분기 지점과 조건은 무엇인가?
- 재귀 호출할 것인가, 루프를 돌 것인가?
- 루프는 무엇을 기준으로 돌고, 탈출 조건은 무엇인가?
- 루프 탈출 후 남아있는 흔적은 어떻게 관리할 것인가?
- 재귀 호출은 꼬리 재귀인가, 아닌가?
- 분리된 모듈들 중 무엇이 블랙 박스이고, 무엇이 아닌가?

설계시 고려 사항

설계에 따라 도구 선택하기 vs. 도구에 따라 설계 다르게 하기

- 도구: 실행 플랫폼, 프로그래밍 언어 등
- 이 수업에서는 대부분 후자로 진행

제어 중심으로 생각하기 vs. 자료 중심으로 생각하기

- 언어와 주제마다 다르지만, 대체로 자료 중심으로 생각하는 게 유리한 경우가 많음
- 객체 지향 개념은 이 두 관점을 자료 중심으로 복합한 것

절차형 설계 vs. 함수형 설계 vs. 논리형 설계

- 이 수업에서 거의 모든 팀들은 (객체 지향이 가미될 수도 있는) 절차형 설계를 하게 됨
- 일부 팀들은 (언어에 따라) '약간의' 함수형 설계도 가능

계층화: 상위 계층 설계 vs. 하위 계층 설계

- 상위 계층일수록 변수의 유효범위scope 와 각 '박스'들의 인터페이스에 더 주목
- 하위 계층일수록 흐름도와 변수 값 변화에 더 주목

(이 과목에서) 설계 밀도 [1/2]

논리적 구조만 설계

- 설계 문서에 '코딩'하는 (즉, 의사코드pseudocode 를 작성하는) 것이 **아님**
- **인터페이스**를 확실히 정해서 명시할 것
  - 각 변수/멤버의 타입과 스코프
  - 함수/메서드의 인자들의 타입과 의미 (어떤 내용의 값이 전달되는가)
  - 함수/메서드의 리턴 타입과 의미
  - 함수/메서드의 부수 작용side-effects (예: 포인터인자/전역변수/멤버 값 변경, 파일 쓰기 등)
- 전역 변수, 함수, 클래스, 인스턴스, 루프, 조건문들
- 데이터 파일의 구조/의미
  - 사용자로부터 숨겨진 (사용자가 못 건드리는) 파일도 설계해야 함
  - CSV/JSON 등 형식이 정해져있어도 세부 사항 (필드 수와 의미, 노드/브랜치 수와 의미 등) 설계 필요
- 극히 단순한 세부 사항은 자연어로 서술 가능 (예: "배열 값 `a[0]` ~ `a[99]` 중 가장 큰 값을 `x`에 대입")
- 이름을 미리 확정할 필요 없는 임시 독립 변수는 이름 없이 서술 가능 (예: "카운터 변수")

(이 과목에서) 설계 밀도 [2/2]

[필수] 설계자 스스로 (기획서와) 설계 문서를 보고 구현할 때, **그제서야 결정해야할 요소가 없어야 함!**
[권장] 다른 개발자가 (기획서와) 설계 문서를 보고 구현할 때, **그제서야 결정해야할 요소가 없는 게 좋음!**

- 나쁜 예: “필요한 ◯◯ 값을 (지금 구현중인) 함수의 인자로 받을까, 전역 변수 값을 읽을까?”
- 나쁜 예: “결과를 정수로 리턴할까, 부동소수점 수로 리턴할까?”
- 나쁜 예: “이 함수의 ◯번째 인자가 ◯◯ 조건에 맞는지 내가 (내부에서) 판정하고 분지해야하나, 밖에서 이미 조건에 맞다고 확인된 경우에만 인자로 들어오나?”

(이 과목에서) 순서도

순서도가 필수는 아니지만, (본인들이 구현시 헷갈리지 않기 위해서라도) 필요한 경우에는 작성 권장

- (벡터 이미지를 권장하지만) 비트맵 이미지로 만들거나 손으로 그려도 됨
- 순서도 노드 내부에는 자연어 설명을 최소화하고 실제 이름과 값 사용 권장
- 해당 순서도에서 사용되는 중요 인자들과 외부 (전역 등) 변수들의 타입과 의미 명시

순서도 기호는 통상적인 관례와 달라도 됨. **단**:

- 시작과 끝을 반드시 (가능한 한 동그라미로) 표시할 것
- 간선$^{\text{edge}}$에 방향(화살촉)을 반드시 표시할 것
- 하나의 노드에서 나가는$^{\text{outgoing}}$ 간선이 둘 이상인 (조건분기인) 경우
  - 간선 시작부분에 표찰$^{\text{label}}$을 반드시 표시
  - 모든 나가는 간선들의 표찰들의 합집합은 해당 노드에서 판정 가능한 경우들의 전체 집합이어야 함

(이 과목에서) 고려 사항

전기프1과의 공통점:

- **무결성**: 설계에 빠진 부분을 구현시에 채워넣기 위해 새로 고민하게 만들면 안됨
- **무모순성**: 구현 불가능한 자체 모순이 있어서는 안됨
- **명확성**: 누가 읽어도 달리 해석할 여지가 없도록 설계
  - 누가 구현해도 (코드는 다르더라도) 결과물이 항상 같은 동작을 해야 함
- **기획 부합성**: 기획이 잘못된 (불가능, 누락 등) 경우, 반드시 기획서부터 수정하고 이를 설계에 반영
- 설계 단계에서 모든 팀원들이 예외없이 역할(부분)을 나눠맡아 설계 수행 권장

전기프1과의 차이점:

- **확장성**: 추후 설계 변경이 용이하도록 미리 대비할 것
- 구현을 미리 조금씩 병행해보면서 설계해도 됨
- 설계시 단순 변심으로 꼭 바꾸고 싶은 초기 기획 요소가 있으면, (강사 확인 후) 바꿔도 됨

[예시] 소인수분해: 초기 기획 내용 vs. 요구사항 변경 가능성

프로그램이 실행되면 주 프롬프트를 띄우고, 입력의 좌우 공백들을 제거한 문자열이 2 이상의 정수 한 개가:

- 아니면 오류 메세지를 출력
- 맞으면 그 수를 소인수분해한 결과(소인수)들을 오름차순으로 나열하여 출력

한 후 종료합니다. 예를 들어, 다음과 같이 작동합니다:

```
C:\> primefac Enter
PrimeFac >>> AKMU Enter
AKMU 는(은) 소인수분해할 수 없습니다!
C:\> primefac Enter
PrimeFac >>> ␣␣␣␣␣360␣␣ Enter
360  =  2 x 2 x 2 x 3 x 3 x 5
C:\>
```

요구자가 추후 다음과 같이 요구할 수도 있음을 유념:

- 경우에 따라 오류 메세지를 다르게 표시해주세요: 인자가 없는 (빈 문자열 한 개인) 경우, 둘 이상인 경우, 하나인데 정수가 아닌 경우, 정수 하나인데 2보다 작은 경우
- 결과/오류 출력 후 주 프롬프트를 다시 띄우고, 입력이 정확히 “quit”일 때에만 종료하게 해주세요.
- 같은 소인수들을 제곱 형태로 묶어서 (예: “2^3 x 3^2 x 5”) 표시해주세요.
- 소인수분해 결과들(소인수들)을 내림차순으로 표시해주세요.

[예시] 소인수분해: 주의사항 (미리 기능을 추가하는 게아님!)

앞 페이지와 같이 초기 기획해놓고 다음과 같이 동작하도록 설계하면 (**기획 미부합**이어서) **안 됨!**

```
C:\> primefac Enter
PrimeFac: 2 이상의 정수 입력 >>> ␣␣␣␣␣360␣␣ Enter
PrimeFac: 출력 형식 선택 - 오름차순(A)/내림차순(D) >>> D Enter
360  =  5 x 3 x 3 x 2 x 2 x 2
C:\>
```

반드시 '겉으로 보기에는' 기획대로**만** 동작하도록 설계해야 함!

- 확장성 대비는 어디까지나 **내부적으로만, 조용히** 해두는 것!
- 추후 '설계를 많이 바꾸지 않고도' 요구사항 변경에 대응할 수 있게 설계하는 것

[예시] 정렬 함수: 확장 미대비

(Python 기준으로) 정렬 함수는 다음과 같이 간단히 설계할 수 있고:

- 필수 매개변수 #1 (s): 수$^{\text{number}}$ 들의 시퀀스
- 리턴값: 수들의 리스트
- 인자 s가 비어있으면 빈 리스트를 리턴, 비어있지 않으면:
  - s의 첫번째 원소보다 작거나 같은 원소들만 골라낸 리스트를 (재귀적으로) “정렬”한 결과 리스트
  - s의 첫번째 원소 하나만 들어있는 (길이가 1인) 리스트
  - s의 첫번째 원소보다 큰 원소들만 골라낸 리스트를 (재귀적으로) “정렬”한 결과 리스트

  를 순서대로 이어붙인 리스트를 리턴

추후 구현시 다음과 같이 간단히 구현되지만:

```
def sort(s):
    if not s: return []
    else: return sort([e for e in s[1:] if e <= s[0]]) + \
                 seq[:1] + \
                 sort([e for e in s[1:] if e >  s[0]])
```

만일 요구사항 변경 때문에 내림차순이나 다른 다양한 정렬 기준들이 필요해지면?

[예시] 정렬 함수: 확장 대비

비교 함수를 인자로 (안 받을 수도 있지만) 받을 수도 있게 준비:

- 필수 매개변수 #1 (s): **임의의 t 타입** 원소들의 시퀀스
- **선택 매개변수 #2 (c):**
  - 만일 있으면, $t \times t \to$ `bool` **타입의 이항 함수**
  - 만일 없으면, 자동적으로 오름차순 산술 대소 비교 (<=)
- 리턴값: **(바로 그) t 타입** 원소들의 리스트
- (계산 흐름은 앞 페이지와 같되, 비교에 산술 대소 비교 대신 이항 함수 c 사용)

```
def sort(s, c=(lambda x, y: x <= y)):
    if not s: return []
    else: return sort([e for e in s[1:] if     c(e, s[0])], c) + \
                  seq[:1] + \
                  sort([e for e in s[1:] if not c(e, s[0])], c)
```

```
>>> sort([(5,1),(7,3),(4,5),(1,6),(2,3),(8,7)]) Enter
[(1, 6), (2, 3), (4, 5), (5, 1), (7, 3), (8, 7)]
>>> sort([(5,1),(7,3),(4,5),(1,6),(2,3),(8,7)], (lambda x, y: x[0]*y[1] <= x[1]*y[0])) Enter
[(1, 6), (2, 3), (4, 5), (8, 7), (7, 3), (5, 1)]
```

확장성을 위한 고려 사항들

왼쪽보다 오른쪽을 우선적으로 고려해볼 것:

- 크기가 정적으로 고정된 배열 vs. (당장 필요없어 보여도) **연결 리스트**
- 통짜 흐름 vs. (당장 필요없어 보여도) **함수**로 분리
- 함수 내부 변수에 고정된 값 대입 vs. (당장 필요없어 보여도) 함수 **매개변수**로 받기
- 단일 클래스로 구성 vs. (당장 필요없어 보여도) **상/하위 클래스**로 분리

기타:

- 전역 (불변) 상수constant 적극 활용
- 전역 변수는 일반적으로 (충돌 위험 때문에) 지양되지만, 확장성 관점에서는 필요할 때가 있음
- 인터페이스 대충 정하지 말 것: 무슨 일(변경)이 생길 수 있을지 상상력 발휘 필요

eCampus 팀프로젝트: 《초기 설계 문서》 제출

일반 사항: **PDF 문서** 형식, **압축 금지** 등은 《초기 기획서 원판》 때와 동일

- 기한: **10/9(월) 오전 11시 직전**까지 (지각 제출은 10/9(월) 저녁 7시 직전까지 허용)

순서도가 있을 경우:

- 모든 순서도들은 **문서 내부에 그림 요소로 삽입**하기를 강력히 권장
- (권장하진 않지만) 부득이 순서도들을 별도 파일(들)로 제출할 경우, **파일명**을:
  - 설계 문서 파일과 순서도 파일을 파일명만으로 빨리 구별할 수 있게 정할 것
  - 순서도 파일들이 여러개일 경우, 각각 파일명에 '무엇에 관한 순서도인지' 간략히 명시

설계 중 초기 기획서 수정

대원칙: 설계시 마음이 바뀌면, **기획부터 수정해놓고** 기획대로 설계!

초기 기획서 원판대로 설계할 수 없거나 기획 내용을 일부 바꾸고 싶을 경우, 초기 기획서 수정판 작성/제출:

- 수정판은 원판과 별도의 항목에 제출하며, 몇 번이든 재제출할 수 있음. (수정 1판, 수정 2판, …)
- 수정할 경우, '초기 설계'는 당연히 초기 기획서 수정판을 기준으로 설계해야 함
- 수정할 필요(나 의사)가 없으면, 제출하지 않아도 됨

미리 알아둘 점: **7주차 이후** ~ 중간발표 전에《1차 요구사항 분석서》라는 이름으로 다시 수정하게 됨

- 요구사항에 조금이라도 관련된 요소들은 반드시 요구사항에 부합하게 기재해야 함 (임의 변경 불가)
- 요구사항과 전혀 관련 없는 요소들은 스스로 판단하여 수정 가능

기획서 수정에 대한 평가 (감점) 원칙

틀린 (즉, 모순이나 누락이 있는) 부분을:

- 중간 발표 전까지 수정하지 않고 그대로 두면 구현 불가/곤란 혹은 기획 미부합: **크게 감점**
- 변심으로 (틀린지 결국 모른 채로) 빼거나 뒤엎으면 **중간 정도 감점**
- (7주차 이후에) 요구사항 변경에 의해 (틀린지 결국 모른 채로) 빼거나 뒤엎으면 **약간 감점**
- 중간 발표 전까지 스스로 발견해서 수정하면 **아주 아주 살짝만** 감점
  - 7주차 이후 요구사항 분석서 작성시 발견해서 수정해도 여기에 해당
  - '어차피 나중에 수정 기회가 있으니 지금 원판은 대충 쓰자' 하지 않도록 유도하는 취지

맞는 (즉, 모순/누락 문제가 없어서 원판대로 설계할 수 있는) 부분일 경우:

- 단순 오타나 표현 실수로 원래 의도와 다르게 쓰여였던 부분은 (강사 확인 없이, 감점 없이) 수정 가능
- 아주 작은 지협적 요소를 살짝 바꾸는 정도는 임의로 (강사 확인 없이, 감점 없이) 수정 가능
- 주요 기능들 중 하나를 크게 바꾸고 싶을 경우, 사전에 강사에게 (훈련 적합성에 관해) 문의!
  - '주제명 및 요약' 제출시 '요약'으로서 기재했던 기능 요소들 중 하나가 달라지는 경우를 뜻함
- 구현 난이도 문제도 사전에 강사에게 문의! (사안에 따라 감점이—대체로 없지만—다름)

기획서 수정판 작성: 기본 원칙

기획서 원판의 복사본을 만들어서, 이를 기초로 수정할 곳만 수정

수정판만 봐도 **'원판[2]과 무엇이 다른지' 한 눈에 알 수 있게** 작성

- 문서 첫머리 및 파일명에 수정판 **판번** 명시
- 문서 초반에 (원판으로부터의) **변경 내역changelog 요약**
- 내용 수정(삭제/추가/변경)시 원판 내용을 남겨둔 채로 수정하고 **변경 전후 차이가 잘 구별되게 표시**

변경 내역도 없고 충분한 표시도 없이 그냥 수정하면 좀 더 크게 감점됨

- 단, 중간 발표시 '깜빡 실수로 표식 없이 수정했던 사항'들을 보고할 기회 있음
- (가능성은 극히 낮지만) 고의로 슬그머니 잠수함 패치한 '물증'이 있으면 부정행위로 간주
  - 전기프2에서는 (감점 기준상) 이럴 필요도 거의 없음

[2] 여기서 "원판"이란 9/27(수) 오전까지 제출했던 문서임. (현재 《수정 n판》 바로 직전의 《수정 n-1판》이 아님.)

기획서 수정판 작성: 판번 및 변경 내역 요약

판번: 수정판 (재)제출 횟수

- 기획서 초반(제목 근처)의 “초기 기획서 원판” 문자열을 “**초기 기획서 수정 ◯판**” 으로 변경

변경 내역 요약: ‘원판’과의 차이점들

- 기획서 초반(목차나 개요보다 앞)에 간략히 요약/정리
- 표제 번호는 필수, 내용은 축약
  - [3 절] 사용 흐름도 ‘검색 후 나가는 간선’ 변경
  - [4.2.2 절] 입력 문자열 조건 일부 (빈 문자열) 삭제
  - [5.1.3 절] 정수 결과값 최대 범위 조건 추가 (99 이하)
  - [5.1.5 절] 날짜 최대 허용 범위 변경 (9999년 → 2037년)
- 각 항목이 ‘수정 몇 판에서 변경된 항목인지’는 표시할 필요 없음[3] (항상 원판과 최종 수정판만 비교)

[3] 이 과목이 아닌 일반적인 상황에서는, 수정 k판이 등장한 일시가 언제고, 그 수정 k판이 수정 $k-1$판과 다른 점이 무엇인지를 각각 나눠서 정리하는 것이 보편적임

기획서 수정판 작성: 내용 (글) 수정 방법

글 삭제/추가/변경 방식:

- 원판에 있던 글을 **삭제할 때**: 실제로 지우지 말고, 삭제한 글임을 **흐린 색 글씨나 눈에 잘 띄는 삭선**으로 (혹은 둘 다 이용해서) 표시

  … 입력 문자열이 ~~빈 문자열이고~~ 최소 한 개 이상의 공백을 포함하면…

- 원판에 없던 글을 **추가할 때**: 추가된 글을 **눈에 잘 띄고 원판에서 안 쓰던 색** 글씨로 표시

  … 해석된 값이 0 이상이고 99 이하의 정수일 경우…

- 원판의 글을 **변경할 때**: 실제로 변경하지 말고,
  - 변경 전의 (원판의) 글을 '삭제된 글'로 표시하고
  - 그 바로 옆이나 아래에 변경 후의 (수정판의) 글을 '추가된 글'로 나란히 표시

  … 날짜는 최대 ~~9999~~2037년 말일까지 허용…

다소 긴 문단이나 목록, 소절 등을 삭제/추가/변경할 때에도 마찬가지로 위 방식 적용

기획서 수정판 작성: 내용 (그림) 수정 방법

본문 속에 삽입된 사용 흐름도나 mockup 이미지를 수정하는 방식:

- 원판에 있던 그림을 **삭제할 때**: 실제로 지우지 말고, 삭제된 그림임을 알아보기 쉽게 표시
- 원판에 없던 그림을 **추가할 때**: 추가된 그림임을 알아보기 쉽게 (안 쓰던 색 테두리 등으로) 표시
- 원판에 있던 그림을 **변경할 때**:
  - 원판의 그림을 지우지 말고 '삭제된 그림'으로 표시하고
  - 그 바로 옆이나 아래에 새 그림을 '추가된 그림'으로 나란히 표시
  - 그림이 복잡하고 차이점을 찾기 힘들 경우, 근처에 차이점 요약 설명

    … '날짜' 드롭다운 메뉴 옆의 '요일' 드롭다운 메뉴 삭제 …
- 간혹, 변경 전후 차이를 **그림 내부에 직접 구별/표시하기 쉬운** 경우:
  - 지워진 부분(노드/간선/표찰 등)을 흐린색으로 표시
  - 추가된 부분(노드/간선/표찰 등)을 눈에 잘 띄고 원판에서 안 쓰던 색으로 표시
  - 이렇게 구별되게 수정할 경우, 원판의 그림 자체를 (남겨두지 말고) 수정된 새 그림으로 아예 교체

사용 흐름도를 (권장하진 않지만) **별도의 파일**로 제출할 때:

- 수정판 **초반의 '변경 내역'**에 "사용 흐름도의 … 부분 변경"이라고 명시
- 업로드하는 이미지 파일(들)의 **파일명에 수정 판번**을 명시 (제출 안내 참고)

그림 수정 예시: 기존 그림 삭제 표시 + 새 그림 추가

달력의 각 날짜의 색상은 기본적으로 검정색이지만:

- 앞/뒤 달의 날짜들은 회색으로 표시하고
- 공휴일은 빨간색으로 표시하고
- 토요일은 파란색으로 표시~~한다~~하되,

위 세 규칙들 중 둘 이상에 동시에 해당되면 그중 더 먼저 서술된 규칙 하나만 따른다.

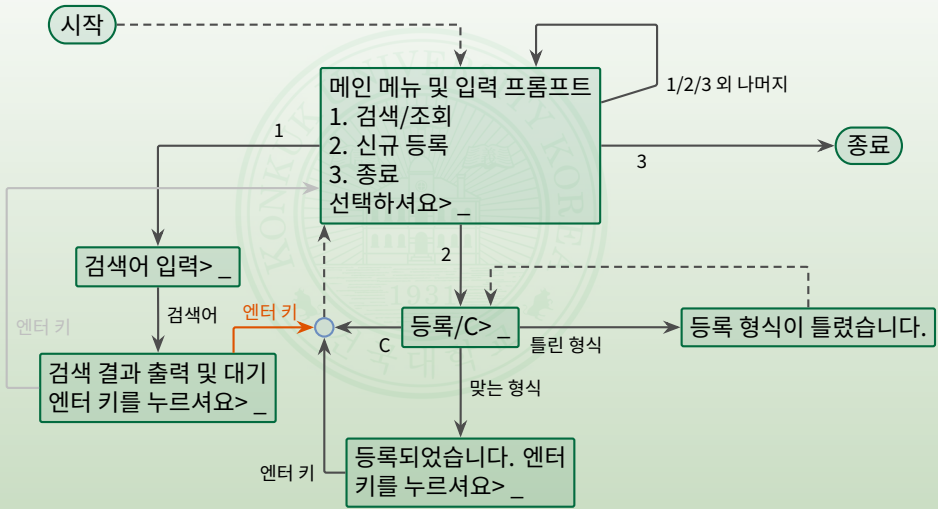
Mo	Tu	We	Th	Fr	Sa	Su
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2

Mo	Tu	We	Th	Fr	Sa	Su
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2

(Mockup 수정: 휴일인 10일을 붉은색으로, 다음달 1일을 회색으로 각각 변경)

그림 수정 예시: 그림 속에 변경 전후 차이를 직접 표시

※ 아래 변경은 내용상 무의미한 변경이며, 어디까지나 예시일 뿐임!



eCampus 팀프로젝트: 《초기 기획서 수정판》 제출

일반 사항: **PDF 문서** 형식, **압축 금지** 등은 《초기 기획서 원판》 때와 동일

- 기한: **10/9(월) 오전 11시 직전**까지 (지각 제출은 10/9(월) 저녁 7시 직전까지 허용)
- 수정판을 만들 필요가 없었던 팀은 제출할 필요 없음

파일명 필수 사항:

- **문서** 파일명에 반드시 판번 명시 (“**수정 ○판**” 형식)
- **흐름도**를 (권장하진 않지만) 부득이 별도 파일(들)로 제출할 경우:
  - 흐름도 파일명에 반드시 흐름도 판번(과 그림 페이지 번호) 명시
    (“**흐름도 수정 ◇판**” 혹은 “**흐름도 수정 ◇판 △번**” 형식)
  - 단, 흐름도 판번(◇)은 문서 판번(○)과 따로 올려도 됨 (서로 똑같이 맞출 필요 없음)
  - 제출한 파일들 중 ‘최종 문서’와 ‘최종 흐름도(들)’를 기준으로 평가함

기타 권장 사항:

- 수정판을 만들 때마다 (혹은 ‘더 수정하지 않을 듯하다’ 싶을 때마다) 그때그때 (재)제출 권장
  - 재제출 횟수는 감점과 전혀 상관 없음 (《수정 1판》을 끝까지 손에 꼭 쥐고 편집할 필요 없음)
- 수정판 신판을 재제출할 때, 기존에 업로드한 수정판 구판들은 지워도 되고 그냥 두어도 됨