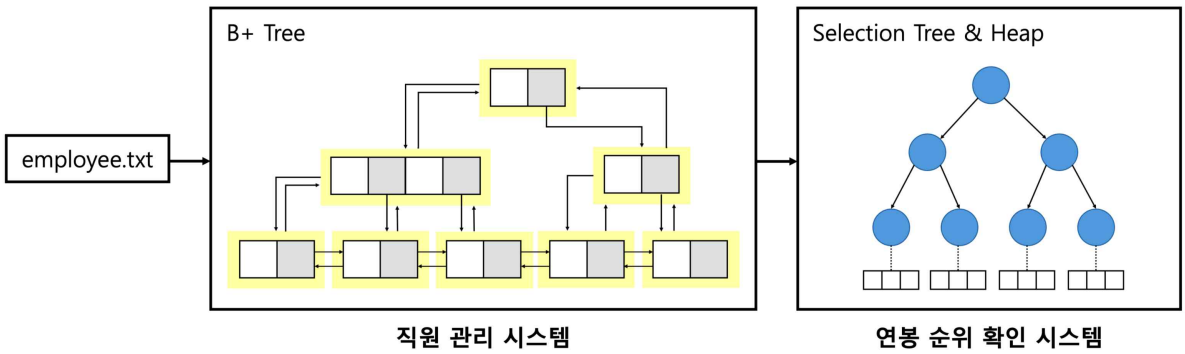
Data Structure Project #2

2025년 10월 13일

Due date: 2025년 11월 9일 일요일 23:59:59까지

본 프로젝트에서는 B+ Tree와 Selection Tree, Heap을 이용하여 회사 직원 관리 프로그램을 구현한다. 직원 관리 프로그램은 이름, 부서코드, 사번, 연봉을 관리하며, 이를 이용해 특정 사원에 대한 정보와 연봉 순위를 제공할 수 있다. B+ Tree를 이용하여 이름 오름차순으로 정렬된 직원을 관리하며, Selection Tree를 이용하여 연봉 순위를 관리한다. Figure 1은 직원 관리 프로그램의 구조이다. 자료구조의 구축 방법과 조건에 대한 자세한 설명은 **program implementation**에서 설명한다.



[figure 1] 직원 관리 프로그램의 예시

□ Program implementation

1) 직원 정보 데이터

- 프로그램은 이름(name), 부서코드(dept_no), 사번(ID), 연봉(annual income) 정보가 저장된 파일 employee.txt를 LOAD 명령어를 통해 읽어 해당 정보를 클래스에 저장한다. employee.txt는 Figure 2와 같이 도서에 대한 정보가 '\t'(탭)로 구분되어 저장되어 있고, 텍스트 파일에 저장된 데이터에는 오류가 없다고 가정한다. 또, 직원 이름은 항상 고유하여 중복인 경우는 없으며 10자 이하로 가정한다. 또한, 파일의 개행 문자는 Line Feed (\n)이다.

```
jaeyong 100     220001  9000
kihun   200     220002  3000
taegwan 300     230001  4000
sunwoo  200     230004  3000
....
```

[Figure 2] 데이터가 저장되어 있는 텍스트 파일의 예

- 부서코드는 표 1과 같이 100~800번까지 있으며, 각 분류 코드에 따라 Selection Tree의

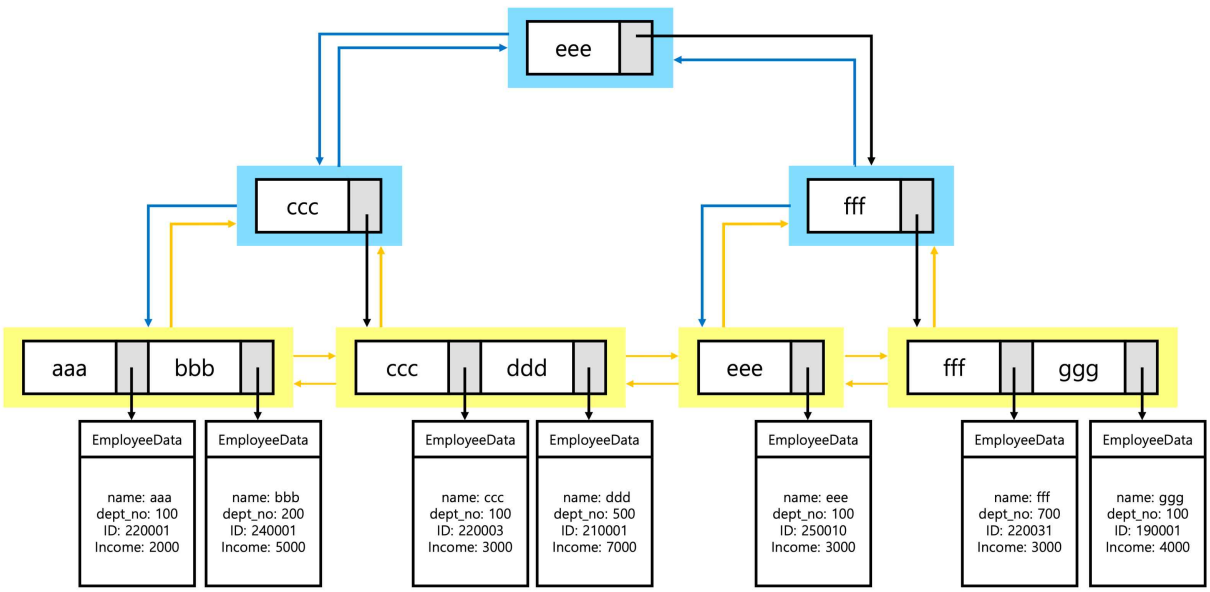
Leaf Node를 구성한다. 이 외의 부서 코드가 input으로 들어오는 경우는 고려하지 않는다.

표 1. 부서 코드 분류

부서 코드 분류
100
200
300
400
500
600
700
800

2) B+ Tree

- B+ tree의 차수는 기본적으로 3이나, 수정이 가능하도록 구현한다. (차수 변경이 구현되지 않은 경우 감점 - 10%)
- 주어진 employee.txt에 저장된 데이터를 읽은 후, 이름을 기준으로 정렬하여 B+ tree에 저장한다.
- ADD_BP 명령어로 추가되는 데이터를 읽은 후, B+ tree에 저장한다. B+ tree에 없으면 node를 새로 추가하며, 존재하는 경우 연봉만 업데이트한다.
- B+ tree에 저장되는 데이터는 EmployeeData class로 선언되어 있으며, 멤버 변수로는 이름, 부서코드, 사번, 연봉이 있다.
- B+ tree는 그림 3의 예시와 같이 이름을 기준으로 하며, 이름은 소문자로만 이루어져 있다.
- B+ tree는 인덱스 노드(BpTreeIndexNode)와 데이터 노드(BpTreeDataNode)로 구성되며, 각 노드 클래스는 B+ tree 노드 클래스(BpTreeNode)를 상속받는다.
- B+ tree의 데이터 노드는 이름, 부서코드, 사번, 연봉을 저장한 EmployeeData를 map 컨테이너 형태로 가지고 있다.

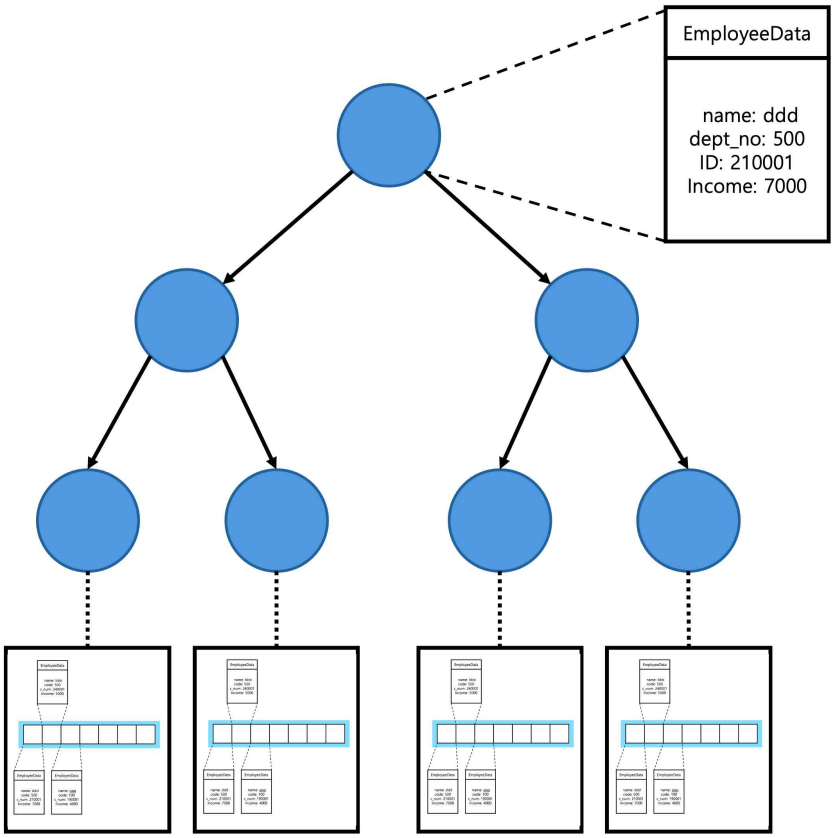


[figure 3] B+ tree 구성 예시

3) Selection Tree

- Selection Tree는 그림 4와 같이 연봉을 기준으로 Max Winner Tree를 구성한다.

- Selection Tree의 run의 개수는 부서코드 개수와 동일하다.
- Selection Tree는 배열이 아닌, 포인터 형태로 구현한다.
- Selection Tree의 내부 node에 저장되는 데이터는 EmployeeData class로 선언되어 있으며, 멤버 변수로는 이름, 부서코드, 사번, 연봉을 갖고 있다.
- Selection Tree의 각 run은 EmployeeHeap class로 선언되며, Heap이 정렬되는 경우 Selection tree로 같이 재정렬된다. EmployeeHeap에 대해서는 다음 절에서 자세히 설명한다.

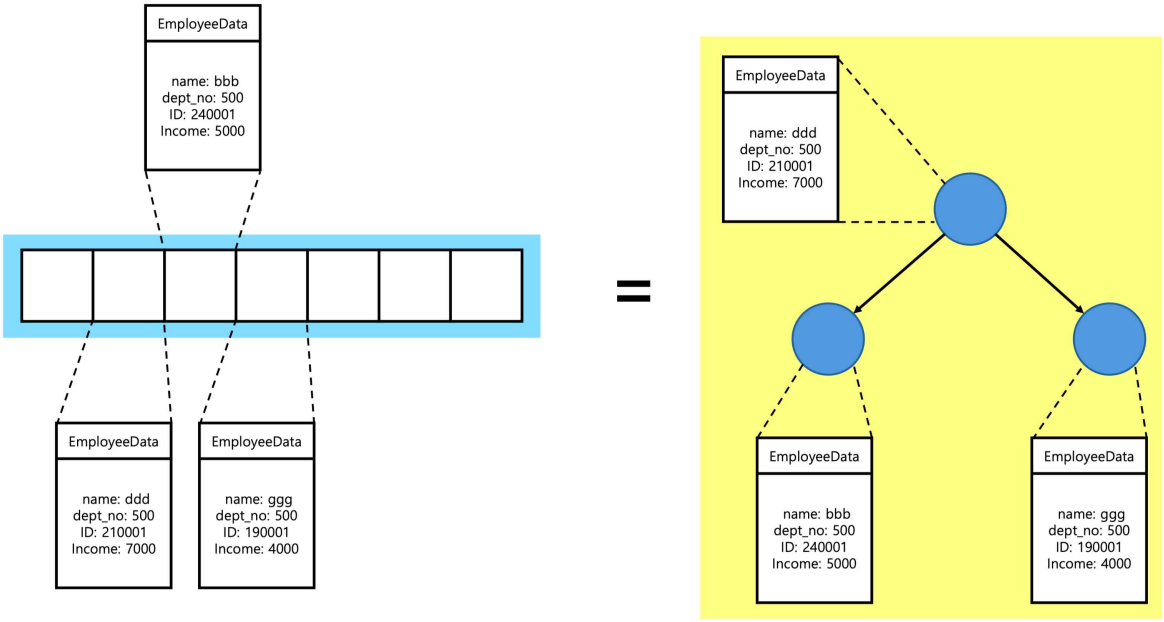


[figure 4] Selection Tree의 예시

4) EmployeeHeap

- EmployeeHeap은 Max Heap으로 그림 5와 같이 연봉을 기준으로 정렬된다.
- Heap은 Linked list 방식이 아닌 포인터 배열로 구현하며, 동적 할당을 통해 구현한다.
- ADD_ST 명령어를 통해 B+ Tree에서 복사해온 데이터로 Heap을 구축하며, 기존에 Heap이 존재하지 않는 경우 새로 구축한다.
  - Heap에 새로운 데이터가 추가할 때는 왼쪽 자식 노드부터 생성한다.
- ADD_ST 명령어를 통해 데이터를 이동해올 때마다 Heap을 재정렬한다.
- Heap을 재정렬할 때, 모든 부모 노드 연봉이 자식 노드의 연봉보다 크거나 같은 경우에 정렬을 완료한다.
  - Heap이 재정렬되는 경우 Selection Tree도 재정렬한다.
- EmployeeHeap에 저장되는 데이터는 EmployeeData class로 선언되어 있으며, 멤버 변수로는 이름, 부서코드, 사번, 연봉을 갖고 있다.

- Figure 5와 같이 배열의 0번째 index는 사용하지 않고, n번째 index의 자식 노드는 2n, 2n+1번째 index가 되도록 구현한다.



[Figure 5] Max Heap의 예시

☐ Functional Requirements

- 출력 포맷은 포맷에 대한 예시일 뿐 실제 출력되는 데이터들과는 차이가 있을 수 있습니다. 이때 명령어에 인자가 존재하는 경우 "\t"(탭)으로 구분합니다.

표 2. 명령어 사용 예 및 기능 설명

명령어	명령어 사용 예 및 기능
LOAD	사용 예) LOAD 텍스트 파일의 데이터 정보를 불러오는 명령어로, 텍스트 파일에 데이터가 존재하는 경우 텍스트 파일을 읽어 B+ tree에 데이터를 저장한다. 만약 텍스트 파일이 존재하지 않거나 자료구조에 이미 데이터가 들어가 있으면 알맞은 에러 코드를 출력한다. 사용되는 텍스트 파일의 이름은 아래와 같으며 파일 이름을 수정하지 않는다. 텍스트 파일: employee.txt ***데이터 조건** - 이름은 10자 미만으로 소문자로 주어지며, 데이터 자체의 오류가 있는 case는 고려하지 않는다. - "\t"를 구분자로 하여 정보가 저장되어 있다. 출력 포맷 예시)

	========LOAD======== Success ======================= ========ERROR======== 100 =======================
ADD_BP	사용 예) ADD_BP name 100 220001 9000 B+ tree에 사원을 직접 추가하기 위한 명령어로, 총 4개의 인자를 추가로 입력한다. 첫 번째 인자부터 사원의 이름, 부서 코드, 사번, 연봉을 나타내며 하나라도 존재하지 않을 시 에러 코드를 출력한다. 주어진 부서 코드 이외의 코드를 입력하는 경우는 없다고 가정하고, 오류가 있는 case는 인자의 수가 4개보다 적은 경우만 가정한다. 출력 포맷 예시) ========ADD_BP======== name/100/220001/9000 ======================= ========ERROR======== 200 =======================
SEARCH_BP	사용 예 1) SEARCH_BP name 사용 예 2) SEARCH_BP a c SEARCH_BP 명령어는 B+ tree에 저장된 데이터를 검색하는 명령어로 1개 또는 2개의 인자를 추가로 입력한다. SEARCH_BP의 인자로 1개가 입력되는 경우는 해당 이름의 검색 결과를 출력한다. 인자가 (시작 단어, 끝 단어)로 2개가 입력되는 경우 범위 검색의 결과를 출력한다. 이때 소문자만을 입력받는다. 예를 들어, 인자로 a와 c가 입력되는 경우 a로 시작하는 사원부터 c로 시작하는 사원을 모두 출력하라는 의미이다. 즉, 대소관계로 나타냈을 때, a <= ? <= c로 나타낸다. 인자를 입력하지 않은 경우, B+ tree에 데이터가 없는 경우, 검색하는 이름의 사원이 없는 경우 에러 코드를 출력한다. 출력 포맷 예시) 1) SEARCH_BP name ========SEARCH_BP======== name/100/220001/9000 =======================

	2) SEARCH_BP a e ========SEARCH_BP======== alice/100/230004/3000 bob/800/230007/4000 cristiano/200/200002/2500 eric/300/210009/5000 ======================== ========ERROR======== 300 ========================
PRINT_BP	사용 예) PRINT_BP B+ tree에 저장된 데이터를 이름을 기준으로 오름차순으로 전부 출력한다. B+ tree에 저장된 데이터가 없는 경우 에러 코드를 출력한다. 출력 포맷 예시) ========PRINT_BP======== alice/100/230004/3000 bob/800/230007/4000 cristiano/200/200002/2500 eric/300/210009/5000 florian/500/250004/7000 lionel/300/230003/9000 mohammed/500/180001/8500 ======================== ========ERROR======== 400 ========================
ADD_ST	사용 예 1) ADD_ST dept_no 100 사용 예 2) ADD_ST name florian ADD_ST 명령어는 B+ Tree에 저장되어 있는 사원 데이터를 Selection Tree에 복사해 오는 역할을 하는 명령어로, 2개의 인자를 받는다. 데이터를 불러오는 방식은 2가지인데, 첫 번째는 부서 코드를 통해 불러오는 방식, 두 번째는 사원 이름을 통해 불러오는 방식이다. dept_no를 통해 불러오는 경우, B+ Tree 내 오름차순 맨 앞의 데이터부터 맨 끝의 데이터를 모두 확인해서 삭제를 진행한다. name을 통해 불러오는 경우 SEARCH_BP 명령어를 활용하는 것처럼 IndexNode를

	통해 검색을 진행한다. 명령어를 실행할 때, B+ Tree에서는 데이터가 존재해야 하며, 각 부서 코드에 알맞은 Heap에 삽입되고, Selection Tree의 업데이트가 일어나야 한다. 만약 검색한 데이터가 존재하지 않거나, 알 수 없는 인자를 입력받은 경우에 에러 코드를 출력한다. 출력 포맷 예시) ========ADD_ST======== Success ======================== ========ERROR======== 500 ========================
PRINT_ST	사용 예) PRINT_ST 500 PRINT_ST 명령어는 Selection tree에서 인자로 받은 코드에 해당하는 데이터를 출력하는 명령어로 1개의 인자를 입력한다. 인자는 부서코드이고, 출력은 사원 이름을 기준으로 오름차순으로 출력한다. Selection Tree에 저장된 데이터가 없는 경우, 사원코드에 대응되는 Heap 자료구조가 없는 경우, 잘못된 부서 코드를 입력하는 경우에 에러 코드를 출력한다. 출력 포맷 예시) ========PRINT_ST======== florian/500/250004/7000 mohammed/500/180001/8500 ======================== ========ERROR======== 600 ========================
DELETE	사용 예) DELETE DELETE 명령어는 Selection Tree에서 root node에 저장된 사원을 제거하는 명령어이다. 이때 사원이 제거되어 Heap에 저장된 사원 연봉의 대소관계가 변경되면 Heap을 재정렬한다. 자료구조에 데이터가 존재하지 않을 시 에러 코드를 출력한다.

	출력 포맷 예시) ========DELETE======== Success ======================== ========ERROR======== 700 ========================
EXIT	사용 예) EXIT 프로그램상의 모든 메모리를 해제하며, 프로그램을 종료한다. 오류가 발생하는 상황은 없다. 출력 포맷 예시) ========EXIT======== Success ========================

□ 동작 별 에러 코드

동작	에러 코드
LOAD	100
ADD_BP	200
SEARCH_BP	300
PRINT_BP	400
ADD_ST	500
PRINT_ST	600
DELETE	700
잘못된 명령어	800

□ Requirements in implementation

- ✓ 모든 명령어는 command.txt에 저장하여 순차적으로 읽고 처리한다.
- ✓ 모든 명령어는 반드시 대문자로 입력한다.
- ✓ 명령어에 인자(Parameter)가 모자라거나 필요 이상으로 입력받으면 알맞은 에러 코드를 출력한다.
- ✓ 예외처리에 대해 반드시 에러 코드를 출력한다.

✓ 출력은 “출력 포맷”을 반드시 따라 한다. “=”의 개수는 고려하지 않아도 된다.
✓ log.txt 파일에 출력 결과를 반드시 저장한다.
- log.txt가 이미 존재할 경우 이전 로그는 삭제 후 새로 저장한다.

□ 구현 시 반드시 정의해야하는 Class

✓ BpTreeNode - B+ tree의 노드 클래스
✓ BpTreeIndexNode - B+ tree의 인덱스 노드 클래스
✓ BpTreeDataNode - B+ tree의 데이터 노드 클래스
✓ BpTree - B+ tree 클래스
✓ SelectionTree - Selection tree 클래스
✓ SelectionTreeNode - Selection tree 노드 클래스
✓ EmployeeHeap - 직원 max heap 클래스
✓ EmployeeData - 직원 데이터 클래스
✓ Manager - Manager 클래스
- 다른 클래스들의 동작을 관리하여 프로그램을 전체적으로 조정하는 역할을 수행

□ Files

✓ employee.txt : 프로그램에 추가할 직원 데이터가 저장된 파일
✓ command.txt : 프로그램을 동작시키는 명령어들을 저장하고 있는 파일
✓ log.txt : 프로그램 출력 결과를 모두 저장하고 있는 파일

□ 제한사항 및 구현 시 유의사항

✓ 반드시 제공되는 코드(github 주소 참고)를 이용하여 구현하며 작성된 소스 파일의 이름과 클래스와 함수 이름 및 형태를 임의로 변경하지 않는다.
✓ 클래스의 함수 및 함수 인자, 변수는 자유롭게 추가 구현이 가능하다.
✓ 제시된 Class를 각 기능에 알맞게 모두 사용한다.
✓ 프로그램 구조에 대한 디자인이 최대한 간결하도록 고려하여 설계한다.
✓ 채점 시 코드를 수정해야 하는 일이 없도록 한다.
✓ 주석은 반드시 영어로 작성한다. (한글로 작성하거나 없으면 감점 - 10%)
✓ 프로그램은 반드시 리눅스(Ubuntu 22.04)에서 동작해야 한다. (컴파일 에러 발생 시 감점 - 50%)
- 제공되는 Makefile을 사용하여 테스트하도록 한다.
✓ **인터넷에서 공유된 코드나 다른 학생이 작성한 코드를 절대 카피하지 않도록 하며, 적발시 전체 프로젝트 0점 처리됨을 명시한다.**

□ 채점 기준

✓ 코드

채점 기준	점수
LOAD 명령어가 정상 동작하는가?	1
ADD_ST 명령어가 정상 동작하는가?	2
SEARCH_BP 명령어가 정상 동작하는가?	3
PRINT_BP 명령어가 정상 동작하는가?	2
ADD_ST 명령어가 정상 동작하는가?	2
PRINT_ST 명령어가 정상 동작하는가?	3
DELETE 명령어가 정상 동작하는가?	2
총합	**15**

– 채점 기준 이외에도 조건 미달 시 감점 (linux 컴파일 에러, 파일 입출력 X, 주석 미흡 등)
- linux 컴파일 에러는 50% 감점, 나머지는 각각 10% 감점
– log.txt 파일에서는 동작하는 것처럼 보이지만, 실제로는 구현이 제대로 되지 않은 경우 감점 (특히, B+ Tree Search)
- 제대로 구현되지 않은 기능마다 최종 점수에서 20점 감점

✓ 보고서

채점 기준	점수
Introduction을 잘 작성하였는가?	1
Flowchart를 잘 작성하였는가?	2.5
Algorithm을 잘 작성하였는가?	3
Result Screen을 잘 작성하였는가?	2.5
Consideration을 잘 작성하였는가?	1
총합	**10**

✓ 최종 점수는 (코드 점수 x 보고서 점수)로 계산됩니다.

□ 제출기한 및 제출방법

✓ 제출기한
- 2025년 11월 9일 일요일 23:59:59까지 클라스(KLAS)에 제출

✓ 제출방법
- 소스코드(Makefile과 텍스트 파일 제외)와 보고서 파일(pdf)을 함께 압축하여 제출
  • 확장자가 .cpp, .h, .pdf가 아닌 파일은 제출하지 않음 (제출시 감점 – 10%)
  • 보고서 파일 확장자가 pdf가 아닐 시 감점 – 10%

✓ 제출 형식
- 학번_DS_project2.tar.gz (ex. 2024123456_DS_project2.tar.gz)
  • 제출 형식을 지키지 않은 경우 감점 – 10%

✓ 보고서 작성 형식 및 제출 방법
- Introduction : 프로젝트 내용에 대한 설명

- Flowchart : 설계한 프로젝트의 플로우차트를 그리고 설명
- Algorithm : 프로젝트에서 사용한 알고리즘의 동작을 설명
- Result Screen : 모든 명령어에 대해 결과화면을 캡처하고 동작을 설명
- Consideration : 고찰 작성
  - 위의 각 항목을 모두 포함하여 작성
  - 보고서 내용은 한글로 작성
  - 보고서에는 소스코드를 포함하지 않음