

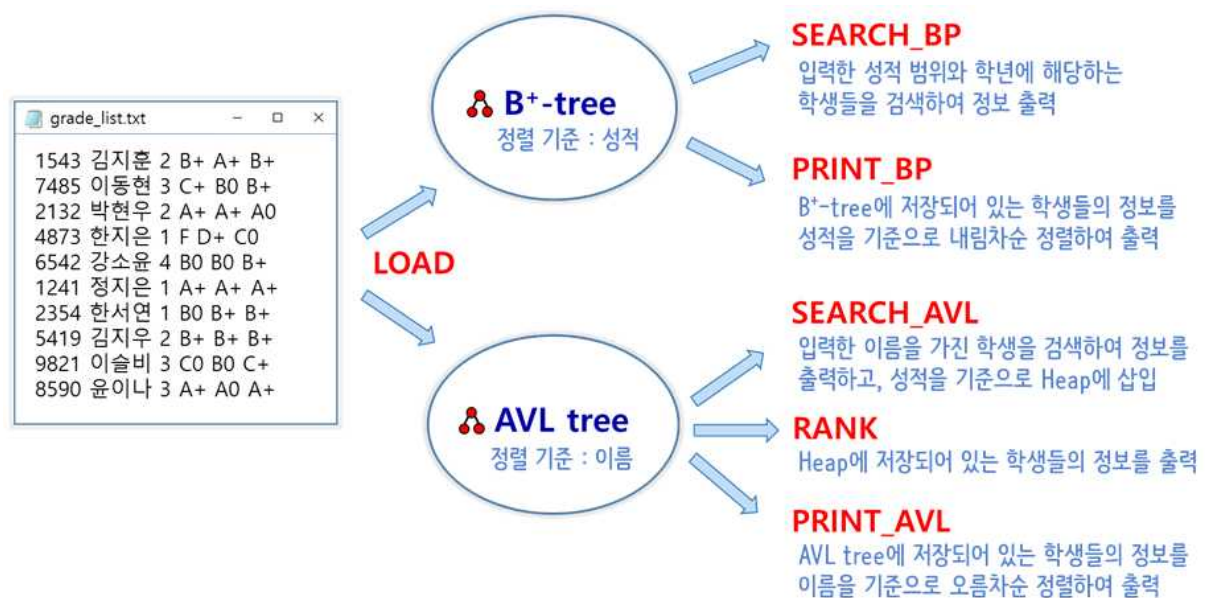
Data Structure Lab. Project #2

2016년 10월 7일

Due date: 2016년 11월 11일 금요일 23:59:59

KW 성적 검색 시스템은 광운대학교 컴퓨터공학과 학생들의 성적 정보를 저장하고, 다양한 기준을 적용하여 효율적으로 검색할 수 있는 시스템이다.

본 프로젝트는 다양한 자료 구조를 활용하여 KW 성적 검색 시스템을 구축하는 것으로, 학생들의 성적 정보를 성적 또는 이름순으로 정렬하여 각각 B⁺-tree와 AVL tree에 저장한다. B⁺-tree는 전체 데이터 출력 기능과 특정 학년의 학생들 중 원하는 성적 범위에 포함되는 학생들의 정보 검색 기능이 있고 AVL tree는 전체 데이터 출력 기능과 특정 학생의 성적 정보 검색 기능이 있다. 이 때, AVL tree는 검색한 학생 정보를 Heap을 이용해 성적순으로 출력하는 기능을 추가적으로 가진다. B⁺-tree와 AVL tree는 삽입 및 탐색 기능만 있을 뿐, 삭제 및 수정 기능은 구현하지 않는다.

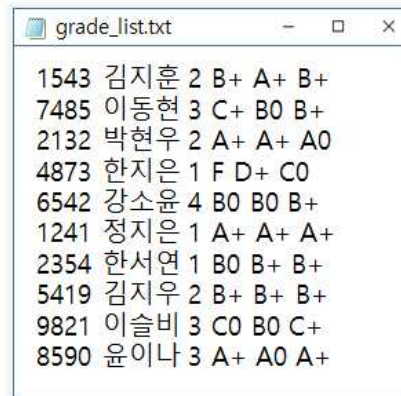


[그림 1] KW 성적 검색 시스템

□ Program implementation

1. 성적 정보 저장

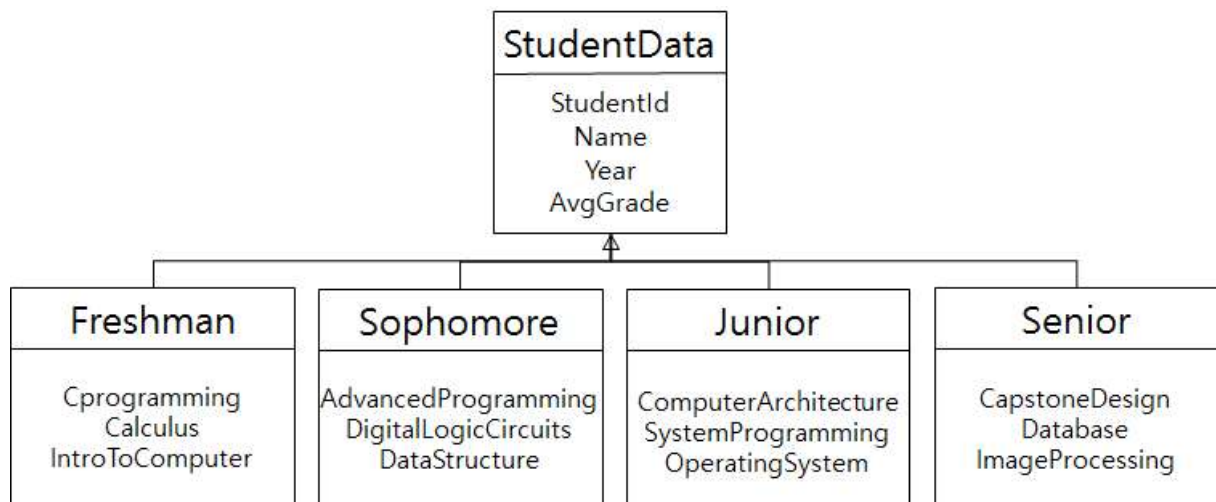
KW 성적 검색 시스템은 학생들의 성적 정보가 저장되어 있는 텍스트 파일(grade_list.txt)을 읽어, 해당 정보를 알맞은 자료구조에 저장한다. 텍스트 파일에는 학번, 이름, 학년, 과목별 성적이 순서대로 저장되어 있다. 학번, 이름, 학년은 '학생 정보 클래스(StudentData)'에 저장하며, 과목별 성적은 학생의 학년에 따라 알맞은 '학년 정보 클래스'에 저장한다. 이 때, 각 학년 정보 클래스에 있는 세 개의 과목에 텍스트 파일에 저장되어 있는 성적순서대로 성적을 저장한다. 학년 정보 클래스는 1학년(Freshman), 2학년(Sophomore), 3학년(Junior), 4학년(Senior) 클래스가 존재하며, 학생 정보 클래스를 상속받는다. 모든 학생들은 세 과목의 성적을 가지고 있으며, 세 과목의 평균 성적을 계산하여 학생 정보 클래스에 저장한다. 평균 성적은 세 과목의 성적을 모두 더한 뒤, 3으로 나누어 계산하는데, 항상 소수점 둘째 자리까지 반올림하여 저장한다. 텍스트 파일에서 중복된 학번의 입력은 없다고 가정한다. 학생들의 성적 정보가 저장되어 있는 텍스트 파일의 예시와 학생 정보 클래스와 각 학년 클래스들의 관계는 다음과 같다. 각 변수에 대한 자세한 내용은 '구현 시 반드시 정의해야 하는 Class 및 멤버 변수'에 작성되어 있다.



```

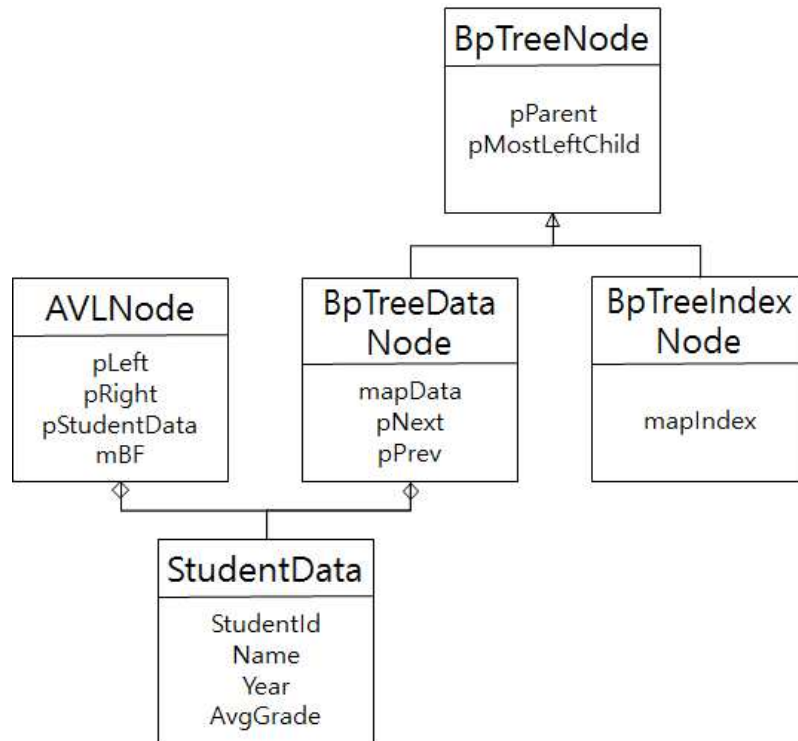
1543 김지훈 2 B+ A+ B+
7485 이동현 3 C+ B0 B+
2132 박현우 2 A+ A+ A0
4873 한지은 1 F D+ C0
6542 강소윤 4 B0 B0 B+
1241 정지은 1 A+ A+ A+
2354 한서연 1 B0 B+ B+
5419 김지우 2 B+ B+ B+
9821 이슬비 3 C0 B0 C+
8590 윤이나 3 A+ A0 A+
  
```

[그림 2] 학생들의 성적 정보가 저장되어 있는 텍스트 파일의 예



[그림 3] 학생 정보 클래스와 각 학년 클래스들의 관계

학생들의 성적 정보는 학생 정보 클래스를 이용한 객체로 저장되고 B⁺-tree와 AVL tree를 구성하는 노드들은 해당 객체 정보를 가진다. B⁺-tree와 AVL tree는 서로 다른 노드를 이용하여 구성되며 학생 정보 클래스와 B⁺-tree 노드 클래스, AVL tree 노드 클래스의 관계는 다음과 같다. 각 변수에 대한 자세한 내용은 '구현 시 반드시 정의해야하는 Class 및 멤버 변수'에 작성되어 있다.

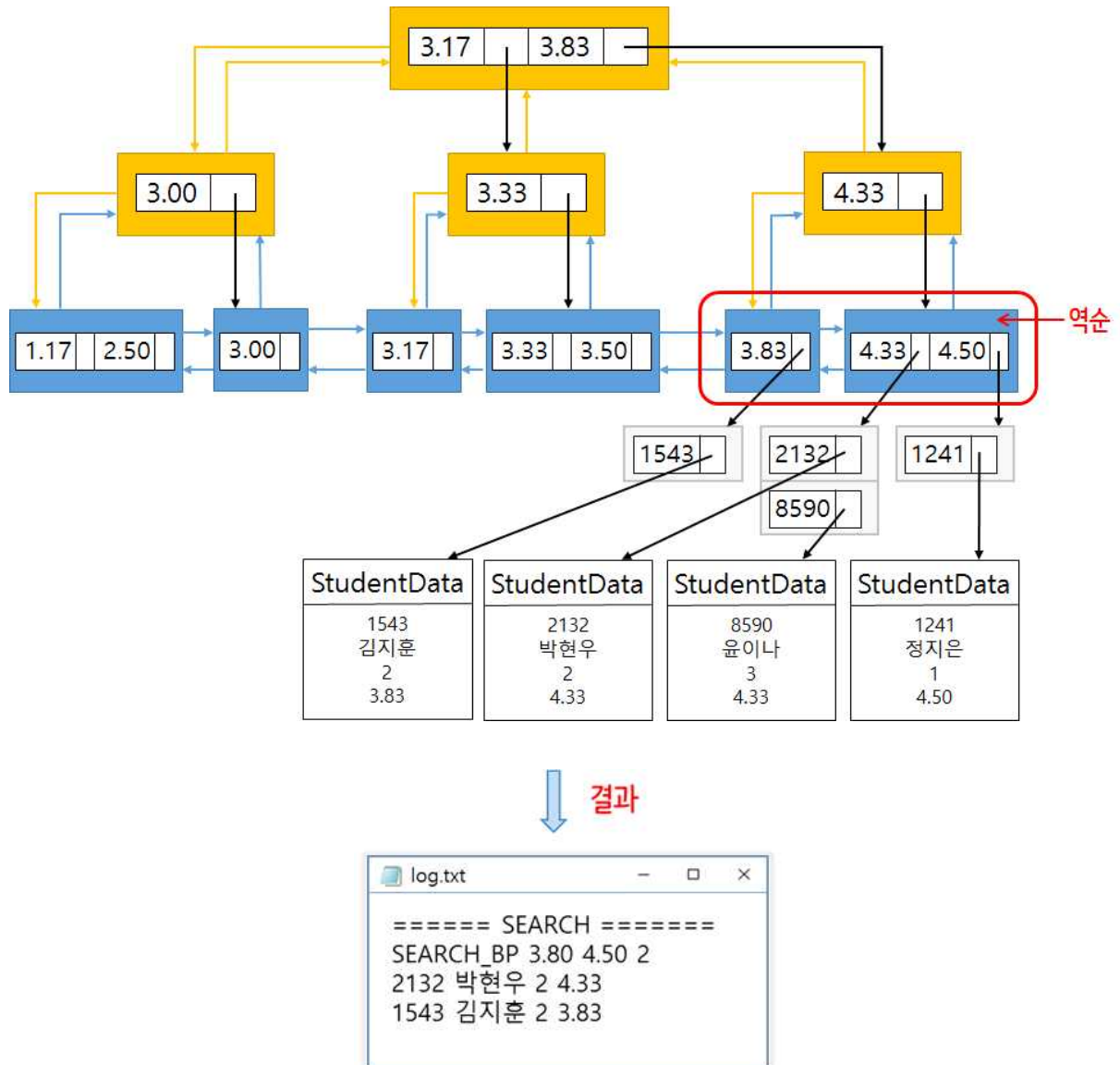


[그림 4] 각 트리의 노드 클래스와 학생 정보 클래스의 관계

B⁺-tree는 성적을 기준으로 학생들의 객체 정보를 저장하고 있으며, 성적이 같은 경우 학번의 오름차순으로 정렬한다. B⁺-tree(BpTree)는 인덱스 노드(BpTreeIndexNode)와 데이터 노드(BpTreeDataNode)로 구성되며, 각 노드 클래스는 B⁺-tree 노드 클래스(BpTreeNode)를 상속받는다. 데이터 노드는 단말 노드로, 학생들의 객체 정보를 map 컨테이너 형태로 가지고 있다. 인덱스 노드는 비단말 노드로, 인덱스와 자식 노드를 가리킬 포인터를 map 컨테이너 형태로 가지고 있으며 가장 왼쪽 자식을 가리키는 포인터를 따로 가지고 있다. 자세한 노드 클래스 정보는 '구현 시 반드시 정의해야하는 Class 및 멤버 변수'에 작성되어 있다. B⁺-tree의 차수 ORDER(m)는 고정되어 있지 않으며 멤버 변수로 변경이 가능하다. B⁺-tree 구성의 예는 다음과 같다.

2. B⁺-tree 검색

KW 성적 검색 시스템은 B⁺-tree를 통해 특정 학년의 학생들 중 원하는 성적 범위에 포함되는 학생들의 정보를 검색할 수 있다. 검색은 성적을 기준으로 역순으로 탐색하면서 특정 학년에 해당되는 학생 정보를 출력한다. 만약 2학년 학생 중 평점이 3.80이상 4.50이하인 학생을 검색한다면, 그 결과는 다음과 같다.



[그림 7] B⁺-tree 검색의 예

3. AVL tree 검색

KW 성적 검색 시스템은 AVL tree를 통해 특정 이름을 가진 학생의 정보를 검색할 수 있다. 검색을 수행할 때마다 그 결과는 출력함과 동시에 vector로 이루어진 Max-Heap에 삽입한다. Heap의 정렬 기준은 성적(MAX) 및 학번(MIN)을 기준으로 한다. RANK 명령을 수행할 때, Heap에 저장된 모든 학생 정보들이 출력 및 삭제된다.

본 프로젝트에서 Max-heap은 표준 템플릿 라이브러리(STL)를 이용하여 구현한다. 시퀀스 컨테이너인 vector를 이용하며, 아래 멤버 함수를 사용하여 구현한다.

- void make_heap(vector의 시작, vector의 끝) : vector의 모든 원소를 heap에 삽입
- void push_heap(vector의 시작, vector의 끝) : vector의 마지막 원소를 heap에 삽입
- void pop_heap(vector의 시작, vector의 끝) : heap의 가장 위에 있는 원소가 vector의 마지막으로 이동

□ Functional Requirements

KW 성적 검색 시스템은 명령어를 통해 동작하며, 실행할 명령어가 작성되어 있는 텍스트 파일(command.txt)을 읽고 명령어에 따라 순차적으로 동작한다. 모든 명령어는 반드시 대문자로 입력하며, 각 명령어의 사용법과 기능은 다음과 같다.

명령어	명령어 사용법 및 기능
LOAD	<p>사용법) LOAD</p> <ul style="list-style-type: none"> ✓ KW 성적 검색 시스템에 학생들의 성적 정보를 불러오는 명령어로, 학생들의 성적 정보가 저장되어 있는 텍스트 파일(grade_list.txt)을 읽어 B⁺-tree와 AVL tree를 구성한다. ✓ 범위(1000~9999)가 맞지 않는 학번의 입력, 한글이 아닌 이름의 입력, 범위(1~4)가 맞지 않는 학년의 입력, 성적이 아닌 다른 문자열의 입력은 없다고 가정한다. ✓ <u>중복된 학번을 입력하는 경우, 학번/이름/학년/3개의 성적 중 하나라도 입력하지 않은 경우는 없다고 가정한다.</u> ✓ 텍스트 파일이 존재하지 않거나, 트리가 이미 구성되어 있는 경우, 출력 파일(log.txt)에 에러 코드(100)를 출력한다.
SEARCH_BP	<p>사용법) SEARCH_BP 시작범위 끝범위 학년 예시) SEARCH_BP 3.00 3.60 2</p> <ul style="list-style-type: none"> ✓ 입력한 성적 범위에 속하는 학생들 중, 해당 학년의 학생들을 검색하는 명령어로, 성적 범위에 속하는 학생들을 B⁺-tree에서 검색한 후, 해당 학년의 학생 정보를 성적의 내림차순으로 출력 파일(log.txt)에 출력한다. ✓ 성적의 시작 범위와 끝 범위는 항상 소수점 둘째자리까지 입력한다고 가정하며, 범위(0.00~4.50)가 맞지 않는 성적의 입력, 시작 범위가 끝 범위

	<p>보다 큰 경우, 범위(1~4)가 맞지 않는 학년의 입력은 없다고 가정한다.</p> <p>✓ 입력한 조건의 학생이 존재하지 않거나, 시작 범위, 끝 범위, 학년 중 하나라도 입력하지 않은 경우, 출력 파일(log.txt)에 에러 코드(200)를 출력한다.</p>
SEARCH_AVL	<p>사용법) SEARCH_AVL 이름</p> <p>예시) SEARCH_AVL 홍길동</p> <p>✓ 입력한 이름을 가진 학생의 성적 정보를 검색하는 명령어로, 학생의 이름을 AVL tree에서 검색하여 출력 파일(log.txt)에 출력한다.</p> <p>✓ 한글이 아닌 이름의 입력은 없다고 가정한다.</p> <p>✓ 입력한 이름의 학생이 존재하지 않거나, 이름을 입력하지 않은 경우, 출력 파일(log.txt)에 에러 코드(300)를 출력한다.</p>
RANK	<p>사용법) RANK</p> <p>✓ AVL tree에서 검색하였던 학생들을 성적의 내림차순, 성적이 같을 경우 학번의 오름차순으로 출력하는 명령어로, Heap에 삽입되어 있는 학생들의 성적 정보를 삭제하면서 출력 파일(log.txt)에 출력한다.</p> <p>✓ Heap에 저장되어 있는 정보가 없을 경우, 출력 파일(log.txt)에 에러 코드(400)를 출력한다.</p>
PRINT_BP	<p>사용법) PRINT_BP</p> <p>✓ B⁺-tree를 탐색하여, 저장되어 있는 모든 정보를 출력하는 명령어로, 출력 파일(log.txt)에 성적의 내림차순으로 출력한다. 성적이 같을 경우, 학번의 오름차순으로 출력한다.</p> <p>✓ B⁺-tree에 저장되어 있는 정보가 없을 경우, 출력 파일(log.txt)에 에러 코드(500)를 출력한다.</p>
PRINT_AVL	<p>사용법) PRINT_AVL</p> <p>✓ AVL tree를 탐색하여, 저장되어 있는 모든 정보를 출력하는 명령어로, 출력 파일(log.txt)에 학생 이름의 오름차순으로 출력한다.</p> <p>✓ AVL tree에 저장되어 있는 정보가 없을 경우, 출력 파일(log.txt)에 에러 코드(600)를 출력한다.</p>
EXIT	<p>사용법) EXIT</p> <p>✓ KW 성적 검색 시스템을 종료하는 명령어로, 종료 시 할당된 메모리를 모두 해제한다.</p>

□ Error Code

명령어에 인자(parameter)가 모자라거나 필요 이상으로 입력 받을 경우, 또는 각 명령어마다 에러 코드를 출력해야 하는 경우, 출력 파일(log.txt)에 명령어 별로 알맞은 에러 코드를 출력한다. 출력 파일(log.txt)이 이미 존재할 경우, 기존 내용 뒤에 이어서 추가로 출력하여 저장한다. 각 명령어별 에러 코드는 다음과 같다.

명령어	에러 코드
LOAD	100
SEARCH_BP	200
SEARCH_AVL	300
RANK	400
PRINT_BP	500
PRINT_AVL	600

□ Print Format

명령어 동작이 성공하였거나 실패한 경우, 출력 파일(log.txt)에 해당 내용을 지정된 형식에 맞추어 출력한다. 출력 파일(log.txt)이 이미 존재할 경우, 기존 내용 뒤에 이어서 추가로 저장한다. 명령어별 출력 형식은 다음과 같다.

명령어	출력 형식
에러(ERROR)	<pre> ===== ERROR ===== 100 ===== </pre>
LOAD	<pre> ===== LOAD ===== Success ===== </pre>
SEARCH_BP	<pre> ===== SEARCH ===== SEARCH_BP 3.80 4.50 2 2132 박현우 2 4.33 1543 김지훈 2 3.83 ===== </pre>
SEARCH_AVL	<pre> ===== SEARCH ===== SEARCH_AVL 한서연 2354 한서연 1 3.33 ===== </pre>

RANK	<p>===== RANK =====</p> <p>2132 박현우 2 4.33 Advanced Programming : A+ Digital Logic Circuits : A+ Data Structure : A0</p> <p>8590 윤이나 3 4.33 Computer Architecture : A+ System Programming : A0 Operating System : A+</p> <p>6542 강소윤 4 3.17 Capstone Design : B0 Database : B0 Image Processing : B+</p> <p>...</p> <p>=====</p>
PRINT_BP	<p>===== PRINT =====</p> <p>1241 정지은 1 4.50 C Programming : A+ Calculus : A+ Intro To Computer : A+</p> <p>2132 박현우 2 4.33 Advanced Programming : A+ Digital Logic Circuits : A+ Data Structure : A0</p> <p>8590 윤이나 3 4.33 Computer Architecture : A+ System Programming : A0 Operating System : A+</p> <p>...</p> <p>=====</p>
PRINT_AVL	<p>===== PRINT =====</p> <p>6542 강소윤 4 3.17 Capstone Design : B0 Database : B0 Image Processing : B+</p> <p>1543 김지훈 2 3.83 Advanced Programming : B+ Digital Logic Circuits : A+ Data Structure : B+</p> <p>...</p> <p>=====</p>

□ Program Behavior Examples

command.txt
<pre> LOAD SEARCH_BP 3.00 4.50 1 SEARCH_AVL 김하나 SEARCH_AVL 한지은 SEARCH_AVL 박현우 SEARCH_AVL 윤이나 SEARCH_AVL 강소윤 RANK PRINT_BP PRINT_AVL EXIT </pre>
실행 결과 화면 및 log.txt
<pre> ===== LOAD ===== Success ===== ===== SEARCH ===== SEARCH_BP 3.00 4.50 1 1241 정지은 1 4.50 2354 한서연 1 3.33 ===== ===== ERROR ===== 300 ===== ===== SEARCH ===== SEARCH_AVL 한지은 4873 한지은 1 1.17 ===== ===== SEARCH ===== SEARCH_AVL 박현우 2132 박현우 2 4.33 ===== ===== SEARCH ===== SEARCH_AVL 윤이나 8590 윤이나 3 4.33 ===== </pre>

===== SEARCH =====

SEARCH_AVL 강소윤

6542 강소윤 4 3.17

=====

===== RANK =====

2132 박현우 2 4.33

Advanced Programming : A+

Digital Logic Circuits : A+

Data Structure : A0

8590 윤이나 3 4.33

Computer Architecture : A+

System Programming : A0

Operating System : A+

6542 강소윤 4 3.17

Capstone Design : B0

Database : B0

Image Processing : B+

4873 한지은 1 1.17

C Programming : F

Calculus : D+

Intro To Computer : C0

=====

===== PRINT =====

1241 정지은 1 4.50

C Programming : A+

Calculus : A+

Intro To Computer : A+

2132 박현우 2 4.33

Advanced Programming : A+

Digital Logic Circuits : A+

Data Structure : A0

...

=====

===== PRINT =====

6542 강소윤 4 3.17

Capstone Design : B0

Database : B0

Image Processing : B+

1543 김지훈 2 3.83

Advanced Programming : B+

Digital Logic Circuits : A+

Data Structure : B+

...

=====

□ 구현 시 반드시 정의해야하는 Class 및 멤버 변수

1. StudentData : 학생 정보 클래스

항목	내용	비고
StudentId	학번	1000번부터 9999번까지만 있는 것으로 가정
Name	이름	이름은 반드시 한글로 저장한다고 가정
Year	학년	1학년부터 4학년까지만 있는 것으로 가정
AvgGrade	평균 성적	평균 성적을 계산하여 저장

2. Freshman : 1학년 정보 클래스

항목	내용	비고
CProgramming	C프로그래밍 성적	- 종류 : A+, A0, B+, B0, C+, C0, D+, D0, F - 본 표의 항목 순서와 텍스트 파일에 작성된 성적의 과목 순서는 동일
Calculus	대수학 성적	
IntroToComputer	컴퓨터개론 성적	

3. Sophomore : 2학년 정보 클래스

항목	내용	비고
AdvancedProgramming	고급프로그래밍 성적	- 1학년 정보 클래스 내용과 동일
DigitalLogicCircuits	디지털논리회로 성적	
DataStructure	데이터구조 성적	

4. Junior : 3학년 정보 클래스

항목	내용	비고
ComputerArchitecture	컴퓨터구조 성적	- 1학년 정보 클래스 내용과 동일
SystemProgramming	시스템프로그래밍 성적	
OperatingSystem	운영체제 성적	

5. Senior : 4학년 정보 클래스

항목	내용	비고
CapstoneDesign	캡스톤설계 성적	- 1학년 정보 클래스 내용과 동일
Database	데이터베이스 성적	
ImageProcessing	영상처리시스템 성적	

6. BpTreeNode : B⁺-tree의 노드 클래스

항목	내용	비고
pParent	상위 BpTreeNode를 가리킴	
pMostLeftChild	하위 BpTreeNode 중 가장 왼쪽 노드를 가리킴	

7. BptreeIndexNode : B⁺-tree의 인덱스 노드 클래스

항목	내용	비고
mapIndex	인덱스 값을 가지고 있는 map 컨테이너	map<double, BpTreeNode*> (평균 성적, 하위 B ⁺ -Tree 노드*)

8. BptreeDataNode : B⁺-tree의 데이터 노드 클래스

항목	내용	비고
mapData	학생 정보 데이터를 가지고 있는 map 컨테이너	map<double, map<int, StudentData*>> (평균 성적, map(학번, 학생 정보 객체*))
pNext	다음 BpTreeNode를 가리킴	
pPrev	이전 BpTreeNode를 가리킴	

9. AVLNode : AVL tree의 노드 클래스

항목	내용	비고
pRight	오른쪽 하위 노드를 가리킴	
pLeft	왼쪽 하위 노드를 가리킴	
pStudentData	학생 정보 객체를 가리킴	
mBF	balance factor	

10. Bptree : B⁺-tree 클래스

항목	내용	비고
root	B ⁺ -tree의 루트	
order	B ⁺ -tree의 차수(m)	멤버 변수로 변경 가능

11. AVLtree : AVL tree 클래스

항목	내용	비고
root	AVL tree의 루트 노드를 가리킴	

12. Manager : 다른 클래스들의 동작을 관리하여 프로그램을 전체적으로 조정하는 역할을 수행

항목	내용	비고
avl	AVL Tree	
bp	B ⁺ -Tree	

□ Files

- ✓ command.txt : 프로그램을 동작시키는 명령어들을 저장하고 있는 파일
- ✓ grade_list.txt : 학생들의 성적 정보를 저장하고 있는 파일
- ✓ log.txt : 프로그램 출력 결과를 모두 저장하고 있는 파일

□ 제한사항 및 구현 시 유의사항

- ✓ 반드시 제공되는 압축파일(project2.tar.gz)를 이용하여 구현하며 작성된 소스 파일의 이름과 클래스와 함수 이름 및 형태를 절대 임의로 변경하지 않는다.
- ✓ 클래스의 함수 및 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)
- ✓ 프로그램은 반드시 리눅스에서 동작해야한다. (컴파일 에러 발생 시 감점)
 - 제공되는 Makefile을 사용하여 테스트 하도록 한다.
- ✓ 소멸자는 제공한 소스코드를 사용하며, 따로 구현하지 않아도 된다.

□ 제출기한 및 제출방법

1. 소스 코드 제출

- ✓ 제출기한
 - 2016년 11월 11일 금요일 23:59:59 까지 제출
- ✓ 제출방법
 - **개인별 제출**
 - 소스코드(Makefile과 텍스트 파일 제외)를 모두 압축하여 제출
 - 확장자가 .cc, .h가 아닌 파일은 제출하지 않음
 - 1차 프로젝트와 동일한 방법으로 제출
- ✓ 제출 형식
 - 설계수강여부_실습반_학번_버전.tar.gz
 - 설계수강여부: 설계미수강(0), 설계수강(1)
 - 실습반: 화요일(A), 수요일(B), 목요일(C), 금요일(D), 미수강(E)
 - Ex1) 설계수강, 실습 화요일반인 경우 : 1_A_2015722000_ver1.tar.gz
 - Ex2) 설계미수강, 실습 목요일반인 경우 : 0_C_2015722000_ver2.tar.gz

2. 보고서 제출

✓ 제출기한

- 2016년 11월 18일 금요일 23:59:59 까지 제출

✓ 제출방법

- **팀별 제출**
- 보고서 파일을 pdf로 변환하고, zip형식으로 압축하여 제출
 - 보고서 파일 확장자가 pdf가 아닐 시 감점
- 제출 방법 추후 공지

✓ 제출 형식

- 설계수강여부_실습반_학번_버전.zip
- 설계수강여부: 설계미수강(0), 설계수강(1)
- 실습반: 화요일(A), 수요일(B), 목요일(C), 금요일(D), 미수강(E)
- Ex1) 설계수강, 실습 화요일반인 경우 : 1_A_2015722000_ver1.zip
- Ex2) 설계미수강, 실습 목요일반인 경우 : 0_C_2015722000_ver2.zip

✓ 보고서 작성 형식

- 하드카피는 제출하지 않음
- 보고서는 한글로 작성
- 보고서에는 소스코드를 포함하지 않음
- 반드시 포함해야하는 항목
 - Introduction : 프로젝트 내용에 대한 설명
 - Flowchart : 설계한 프로젝트의 플로우차트를 그리고 설명(모든 명령어에 대하여 각각 그릴 것)
 - Algorithm : 프로젝트에서 사용한 알고리즘의 동작을 설명(B⁺-tree 삽입, AVL tree 삽입, Heap push, Heap pop에 대하여 각각 예시를 들어 설명할 것)
 - Result Screen: 모든 명령어에 대하여 각각 결과화면을 캡처하고 동작을 설명
 - Consideration : 고찰 작성

※ 고찰 작성법

- 개인 별로 각각 1page 이상 작성
- 자신이 스스로 프로젝트를 진행하면서 배운 점, 어려웠던 점, 해결 방안 등을 작성
- 팀원들의 소스코드를 검토해보고 배운 점, 고칠 점 등을 구체적 사례를 들어 상세하게 작성 (예: AVLtree class의 Insert()에서 root가 없을 때의 예외처리를 하지 않았다.)
- 한 사람이 모든 팀원들에 대해 각각 작성(예: 팀원이 A, B, C 3명일 때, A는 A 자신의 고찰, 팀원 B에 대한 고찰, 팀원 C에 대한 고찰을 1page 이상 작성해야 함)
- 모여서 소스코드를 검토할 때의 인증샷 포함

□ 퀴즈

- ✓ 일시 : 11월 12일 토요일 오전 11시
- ✓ 범위 : 프로젝트에서 사용되는 알고리즘 전체에 대해 출제 (구현되어 제공된 소스코드 포함)
- ✓ 장소 : 추후 공지