KMU - File Processing Report

01. Adelson-Velskii Landis Tree (AVL)

Common Instruction

- 모든 내용은 **C/C++ 언어**로 작성해야 합니다.
- 입/출력, 메모리 할당/회수, 노드 삭제 로직에 사용되는 라이브러리를 제외한 **모든 표준 라이브러리의 사용은 허용되지 않습니다.**
- 프로그램은 정의되지 않은 행동을 제외하고, **예기치 못한 상황(Segmentation fault, Bus error, Double free, Abort 등)으로 인해 종료되어서는 안됩니다.**
- Heap 에 할당된 모든 메모리는 반드시 올바르게 회수되어야 하며, **메모리 누수는 허용되지 않습니다.**
- 소스 코드 화일 명은 **학번_이름_AVL.c** 혹은 **학번_이름_AVL.cpp** 로 합니다.
- **모든 부정행위는 0 점 처리됩니다.**
- AVL Report 에서 얻을 수 있는 최대 점수는 **100 점**입니다.

생성형 AI

- 생성형 AI 를 아래의 **제약 조건 내**에서 사용할 수 있습니다.
    - 소스 코드 화일(학번_이름_AVL.c 혹은 학번_이름_AVL.cpp)와 함께, 관련 **보고서를 추가로 제출**해야 합니다.
    - 보고서에는 **생성형 AI 종류**를 명시합니다.
    - **모든 입력 프롬프트와 출력 값**을 명시합니다. 즉, 보고서를 통해 최종 코드가 어떻게 생성되었는 지의 과정을 볼 수 있어야 합니다.
    - 보고서 명은 **학번_이름_AVL_생성형 AI 사용보고서.pdf** 로 합니다.
- 생성형 AI 를 사용할 경우 **받을 수 있는 최대 점수는 66 점** 입니다.
- **아래의 항목들은 부정행위로 간주되어 0 점 처리됩니다.**
    - 생성형 AI 사용 보고서를 제출하지 않고, 생성형 AI 로 생성한 코드를 제출한 경우
    - 코드를 구현해달라는 요청의 경우

Mandatory Implementation

1. AVL 의 각 노드를 표현하는 클래스 또는 구조체를 구현하시오.
    - 클래스 또는 구조체 명은 “Node”로 한다.
    - 노드는 다음 멤버 변수를 가진다.
        - key: 해당 노드의 키 값
        - left: 해당 노드의 왼쪽 자식 노드의 주소
        - right: 해당 노드의 오른쪽 자식 노드의 주소
        - height, size, bf 를 제외한 다른 멤버 변수는 **추가, 삭제 및 변경이 금지된다.**
2. AVL 의 삽입 알고리즘을 구현하시오.
    - 단, 수업 슬라이드에서 설명한 아래의 함수를 이용하여 구현해야 한다.
        - insertBST(T, key)
        - checkBalance(T, key)
        - rotateTree(T)
    - 함수는 아래의 프로토타입(템플릿)을 따라야 한다.
        - insertAVL(T, key)
3. AVL 의 삭제 알고리즘을 구현하시오
    - 단,수업 슬라이드에서 설명한 아래의 함수를 이용하여 구현해야 한다.
        - eraseBST(T, key)
        - checkBalance(T, key)
        - rotateTree(T)
    - 함수는 아래의 프로토타입(템플릿)을 따라야 한다.
        - eraseAVL(T, key)
4. 출력에 사용할 트리의 inorder 순회 알고리즘을 구현하시오.
    - 함수는 아래의 프로토타입(템플릿)을 따라야 한다.
        - inorder(T)
5. AVL 의 clear 알고리즘을 구현하시오.
    - 함수는 아래의 프로토타입(템플릿)을 따라야 한다.
        - clear(T)

Input

- 입력은 표준 입력(STDIN)을 통해 주어진다.
- 삽입/삭제를 나타내는 문자와 삽입/삭제 대상이 되는 키 값이 공백을 사이에 두고 여러 줄에 걸쳐 주어진다.
    - 이 때, 삽입 명령은 'i', 삭제 명령은 'd'로 주어진다.
- 입력의 개수는 최대 1,000 개이며, key 값은 int 범위 내의 정수로 주어진다.
    - 입력의 개수는 주어지지 않는다.
- 정의되지 않은 입력은 주어지지 않는다.

Output

- 출력은 표준 출력(STDOUT)을 통해 이루어져야 한다.
- 아래의 각 경우에 대해 정해진 형식에 맞게 출력한다. 명시된 형식에 정확히 일치하지 않는 경우 **0 점 처리된다.**
    - 정상 작동
        - 각 삽입/삭제 명령 처리가 완료된 후, 트리의 inorder 순회 결과를 한 줄에 출력한다.
        - 트리 순회 결과는 방문한 순서대로 공백을 사이에 두고 출력한다.
        - 이 때, 트리 구조의 검증을 위해 **아래와 같은 형식으로 출력한다.**
            - <{left subtree}‘ ’{key}‘ ’{right subtree}>
            - 공백은 “\0x20” 의 공백만 허용되며, **‘ ’에 출력한다.**
            - ‘{var}’에는 해당 칸에 알맞은 값을 출력한다.
            - ‘<’, ‘>’는 실제로 출력해야 할 문자이다.
    - 에러 상황
        - 이미 존재하는 키 값에 대한 삽입 명령이 주어진 경우 아래의 에러 메시지를 출력한다.
            - “i {key}: The key already exists”
        - 없거나 이미 삭제된 키 값에 대해 삭제 명령이 주어진 경우 아래의 에러 메시지를 출력한다.
            - “d {key}: The key does not exist”
        - ‘{var}’ 에는 해당 칸에 알맞은 값을 출력한다.
        - 에러 메시지의 경우 표준 에러(STDERR)로의 출력 또한 허용된다.

예제 입력

```
i 25
i 500
i 25
i 33
i 49
i 17
i 403
i 29
i 105
i 39
i 66
i 305
i 44
i 19
i 441
i 390
i 12
i 81
i 50
i 100
i 999
d 25
d 500
d 25
d 33
d 49
d 17
d 403
d 29
d 105
d 39
d 66
d 305
d 44
d 19
d 441
d 390
d 12
d 81
d 50
```

```
d 100
d 999
```

예제 출력

```
< 25 >
< 25 < 500 >>
i 25: The key already exists
<< 25 > 33 < 500 >>
<< 25 > 33 << 49 > 500 >>
<<< 17 > 25 > 33 << 49 > 500 >>
<<< 17 > 25 > 33 << 49 > 403 < 500 >>>
<<< 17 > 25 < 29 >> 33 << 49 > 403 < 500 >>>
<<< 17 > 25 < 29 >> 33 << 49 < 105 >> 403 < 500 >>>
<<< 17 > 25 < 29 >> 33 <<< 39 > 49 < 105 >> 403 < 500 >>>
<<< 17 > 25 < 29 >> 33 <<< 39 > 49 < 66 >> 105 < 403 < 500 >>>>
<<< 17 > 25 < 29 >> 33 <<< 39 > 49 < 66 >> 105 << 305 > 403 < 500 >>>>
<<<< 17 > 25 < 29 >> 33 < 39 < 44 >>> 49 << 66 > 105 << 305 > 403 < 500 >>>>
<<<< 17 < 19 >> 25 < 29 >> 33 < 39 < 44 >>> 49 << 66 > 105 << 305 > 403 < 500 >>>>
<<<< 17 < 19 >> 25 < 29 >> 33 < 39 < 44 >>> 49 <<< 66 > 105 < 305 >> 403 << 441 > 500
>>>
<<<< 17 < 19 >> 25 < 29 >> 33 < 39 < 44 >>> 49 <<< 66 > 105 < 305 < 390 >>> 403 << 441 >
500 >>>
<<<<< 12 > 17 < 19 >> 25 < 29 >> 33 < 39 < 44 >>> 49 <<< 66 > 105 < 305 < 390 >>> 403 <<
441 > 500 >>>
<<<<< 12 > 17 < 19 >> 25 < 29 >> 33 < 39 < 44 >>> 49 <<< 66 < 81 >> 105 < 305 < 390 >>>
403 << 441 > 500 >>>
<<<<< 12 > 17 < 19 >> 25 < 29 >> 33 < 39 < 44 >>> 49 <<<< 50 > 66 < 81 >> 105 < 305 < 390
>>> 403 << 441 > 500 >>>
<<<<< 12 > 17 < 19 >> 25 < 29 >> 33 < 39 < 44 >>> 49 <<< 50 > 66 < 81 < 100 >>> 105 <<
305 < 390 >> 403 << 441 > 500 >>>>
<<<<< 12 > 17 < 19 >> 25 < 29 >> 33 < 39 < 44 >>> 49 <<< 50 > 66 < 81 < 100 >>> 105 <<
305 < 390 >> 403 << 441 > 500 < 999 >>>>>
<<<<< 12 > 17 > 19 < 29 >> 33 < 39 < 44 >>> 49 <<< 50 > 66 < 81 < 100 >>> 105 << 305 <
390 >> 403 << 441 > 500 < 999 >>>>>
<<<<< 12 > 17 > 19 < 29 >> 33 < 39 < 44 >>> 49 <<< 50 > 66 < 81 < 100 >>> 105 << 305 <
390 >> 403 < 441 < 999 >>>>>
d 25: The key does not exist
<<<<< 12 > 17 > 19 > 29 < 39 < 44 >>> 49 <<< 50 > 66 < 81 < 100 >>> 105 << 305 < 390 >>
403 < 441 < 999 >>>>>
```

<<<<< 12 > 17 > 19 > 29 < 39 < 44 >>> 50 << 66 < 81 < 100 >>> 105 << 305 < 390 >> 403 < 441 < 999 >>>>>
<<<< 12 > 19 > 29 < 39 < 44 >>> 50 << 66 < 81 < 100 >>> 105 << 305 < 390 >> 403 < 441 < 999 >>>>>
<<<< 12 > 19 > 29 < 39 < 44 >>> 50 << 66 < 81 < 100 >>> 105 << 305 > 390 < 441 < 999 >>>>>
<<< 12 > 19 < 39 < 44 >>> 50 << 66 < 81 < 100 >>> 105 << 305 > 390 < 441 < 999 >>>>>
<<< 12 > 19 < 39 < 44 >>> 50 << 66 < 81 < 100 >>> 305 < 390 < 441 < 999 >>>>>
<<<< 12 > 19 < 44 >> 50 < 66 < 81 < 100 >>>> 305 < 390 < 441 < 999 >>>>
<<<< 12 > 19 < 44 >> 50 < 81 < 100 >>> 305 < 390 < 441 < 999 >>>>
<<<< 12 > 19 < 44 >> 50 < 81 >> 100 < 390 < 441 < 999 >>>>
<<<< 12 > 19 > 50 < 81 >> 100 < 390 < 441 < 999 >>>>
<<< 12 > 50 < 81 >> 100 < 390 < 441 < 999 >>>>
<<< 12 > 50 < 81 >> 100 < 390 < 999 >>>
<<< 12 > 50 < 81 >> 100 < 999 >>
<< 50 < 81 >> 100 < 999 >>
<< 50 > 100 < 999 >>
< 100 < 999 >>
< 999 >
// 해당 라인은 마지막 노드가 삭제된 후, 트리의 inorder 순회 출력이 없으며 개행만 발생한 경우입니다.

채점 방식

채점은 여러 테스트 케이스에 대해 수행되며, 각 테스트 케이스에 대한 inorder 출력 값과 에러 메시지 출력 값을 이용합니다.

아래 링크는 여러분의 과제를 도와줄 수 있는 레포트 채점기에 대한 git repository 링크입니다. 아래 채점기의 채점 결과는 여러분의 **실제 과제 점수와 다르다는 것**을 미리 안내드립니다. 버그 리포트 및 컨트리뷰션은 환영합니다.

https://github.com/KMU-File-Processing/File-Processing-Report-Tester.git

※ 레포트에 관련된 모든 질문은 DM, Email 이 아닌 slack 의 **리포트_qna** 채널을 통해서만 가능합니다.