B-Tree

구현내용

B-tree의 각 노드를 표현하는 클래스 또는 구조체를 구현하시오.

- 각 노드는 다음 속성을 필수적으로 가지도록 한다.
  - n: 해당 노드가 가지고 있는 키의 개수
  - K: 해당 노드가 가지고 있는 키들을 담고있는 배열
  - P: 해당 노드가 가지고 있는 포인터들(자식 노드)을 담고있는 배열

B-tree의 삽입 알고리즘 insertBT(T, m, newKey)을 구현하시오.

- 단 알고리즘은 다음 함수를 이용해 구현하시오.
- searchPath(T, m, key, stack)
  - key에 대한 검색을 수행
  - key 발견 여부와 방문한 정점 경로를 담은 stack을 리턴
  - 주어진 stack이 null이 아닌 스택이 주어지면, 해당 스택에 경로를 저장
- insertKey(T, m, x, y, newKey)
  - newKey를 노드 x의 적당한 위치에 삽입
  - y는 x에서 newKey위치 다음 포인터가 가리킬 노드 (자식 노드에서 split되어 새로 만들어진 노드)
- splitNode(T, m, x, y, newKey)
  - newKey를 x에 삽입 후 split을 수행
  - y는 x에서 newKey위치 다음 포인터가 가리킬 노드 (자식 노드에서 split되어 새로 만들어진 노드)
  - split 후 부모 노드에 삽입되어야 할 키값과 분리된 새로운 노드를 반환

B-tree의 삭제 알고리즘 deleteBT(T, m, oldKey)을 구현하시오.

- 단 알고리즘은 다음 함수를 이용해 구현하시오.
- searchPath(T, m, key, stack)
- deleteKey(T, m, x, oldKey)
  - oldKey를 x에서 제거 수행
- bestSibling(T, m, x, y)
  - 노드 x와 해당 노드의 부모 노드 y가 주어졌을 때, x의 best 형제노드를 반환
- redistributeKeys(T, m, x, y, bestSibling)
  - 노드 x와 해당 노드의 부모 노드 y, 그리고 y에서의 best sibling의 인덱스 bestSibling이 주어졌을 때 x와 best sibling 노드간의 키 재분배 수행
- mergeNode(T, m, x, y, bestSibling)

    - 노드 x와 해당 노드의 부모 노드 y, 그리고 y에서의 best sibling의 인덱스 bestSibling이 주어졌을 때 x와 best sibling 노드간의 합병 수행

출력을 위해 B-tree의 inorder 순회 알고리즘을 구현하시오

- inorderBT(T, m)

입력

입력은 텍스트 화일을 통해 주어진다.
삽입 또는 삭제를 나타내는 문자와 삽입 또는 삭제 대상이 되는 키값이 공백을 사이에 두고 여러 줄에 걸쳐 주어진다. 즉, 입력 화일의 한 줄은 삽입 또는 삭제를 나타내는 명령과 정수값으로 구성된다. 이때 삽입 명령은 'i', 삭제 명령은 'd'로 주어진다.

트리 초기화 또는 삽입 또는 삭제를 나타내는 문자와 정수값이 공백을 사이에 두고 여러 줄에 걸쳐 주어진다. 이때, 삽입은 'i', 삭제는 'd'로 주어진다. 삽입 또는 삭제 문자가 주어진 경우, 정수값은 삭제 대상이 되는 키값을 타나낸다.

출력

출력은 반드시 표준출력을 통해 이루어져야 한다.
주어진 삽입/삭제 명령에 대해 먼저 m=3인 경우에 수행한 결과를 출력하고, 다음으로 m=4인 경우에 수행한 결과를 출력한다.
다음 주어진 각 경우에 대해 반드시 주어진 형식에 맞추어 출력한다.

- 이미 존재하는 키값에 대한 삽입 명령이 주어진 경우
    - "i 〈key〉 : The key already exists"을 따옴표를 제외하고 한줄에 출력
    - 〈key〉에는 입력으로 주어진 키값을 출력
- 없거나 이미 삭제된 키값에 대해 삭제 명령이 주어진 경우
    - "d 〈key〉 : The key does not exist"을 따옴표를 제외하고 한줄에 출력
    - 〈key〉에는 입력으로 주어진 키값을 출력
- 각 삽입/삭제 명령에 대한 처리를 마친 뒤
    - 트리 순회 결과를 한줄에 출력
    - 트리 순회 결과는 방문한 순서대로 키값을 공백을 사이에 두고 출력한다.

예제 입출력

예제입력

```
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
```

예제출력

```
25
25 500
i 25 : The key already exists
25 500
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 33 39 44 49 50 66 81 100 105 305 390 403 441 999
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

25
25 500
```

```
i 25 : The key already exists
25 500
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 33 39 44 49 50 66 81 100 105 305 390 403 441 999
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
```

채점 방식

정확성 채점

정확성 채점은 여러 테스트 케이스에 대해 수행되며, 각 테스트 케이스에 대한 inorder 출력값과 에러 메시지 출력값을 이용한다. 이때 inorder 출력의 경우, 내부에서 생성한 트리 구조를 파악하기 위해 키값 뿐만 아니라 탐색 순서까지 출력하도록 코드가 수정된다. 출력값을 바탕으로 채점기가 트리를 생성하고, 삽입/삭제가 완료되는 시점 (inorder를 출력하는 시점)에 자료구조의 성질을 만족하는지 확인하는 방식으로 이루어진다. 따라서 출력을 주어진 형식과 조건에 맞추어야 제대로된 채점을 받을 수 있다.

효율성 채점

효율성 채점은 여러 테스트 케이스에 대해 수행되며, 각 테스트 케이스에 대해 출력을 하지 않도록 코드를 변경했을 때의 실행 시간을 측정하여 자료구조가 의도한 성능을 내는지 여부를 채점한다.

제출물

- Source code
    - 각 함수에 대한 설명을 주석으로 작성 (필수)
    - 함수 뿐만 아니라 소스 코드에도 충분한 주석을 달 것
- 실행환경에 대한 설명
    - 컴파일 명령어, 버전 정보, 운영체제 정보