**Introduction to Databases, Spring 2020**

**Homework #4 (May 27, 2020)**

**Student ID 2016312761**

**Name 여혁수**

Compress ‘main.c’, ‘BTREE.c’, ‘BTREE.h’ and ‘your report’ (this current document file) and submit with the filename ‘HW4\_STUDENT ID.zip’

**NOTE**: You can add/modify functions if you want, but don’t use additional libraries.

**(1)** [**30 pts**] Implement **insertion** and **deletion** operations of B-tree and write the codes. In addition, for given element sequences, show the results together. **(Insertion : 15pts, Deletion 15 pts)**

Definition of B-tree

1. Every node has at most m children (m: B-tree of order).

2. Every non-leaf node (except root) has at least ⌈m/2⌉ children.

3. A non-leaf node with k children contains k−1 keys.

4. All leaves appear in the same level

To implement the B-tree, please refer to the following sites.

- https://en.wikipedia.org/wiki/B-tree#Terminology

- https://www.cs.usfca.edu/~galles/visualization/BTree.html

**(a)** You should fill the implementation code in the B-tree template.

**Answer: Submit your code to i-campus. Don’t write your code here.**

**(b)** Show the B-tree for each case.

**Answer: Show your results. (Drawing or Snapshot)**

1. Max degree = 3

Insert(1, 3, 7, 10, 11, 13, 14, 15, 18, 16, 19, 24, 25, 26)

1. Max degree = 4

Insert(1, 3, 7, 10, 11, 13, 14, 15, 18, 16, 19, 24, 25, 26)

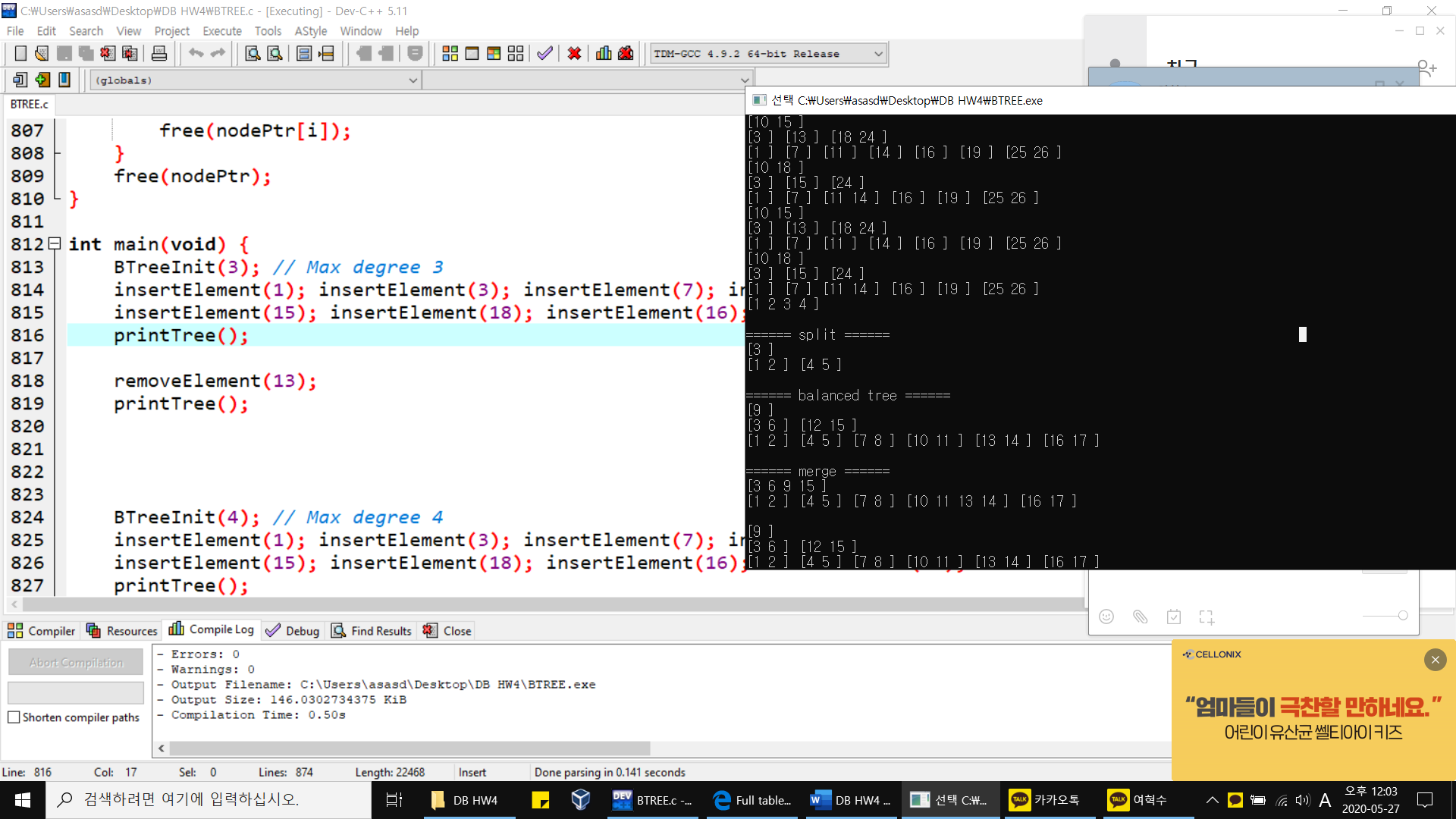
**For Deletion :**

1. Max degree = 3

Insert(1, 3, 7, 10, 11, 13, 14, 15, 18, 16, 19, 24, 25, 26) Remove (13)

1. Max degree = 4

Insert(1, 3, 7, 10, 11, 13, 14, 15, 18, 16, 19, 24, 25, 26) Remove (13)



**(2) [20 pts]** Compare the index scan and full table scan using SQL queries on MySQL. The selectivity of a predicate indicates how many rows from a row set will satisfy the predicate.

Compare the running time between index scan and full table scan according to different data selectivity and draw the graph to compare two scan methods depending on the selectivity. (Fix the total number of rows as 20,000,000). You also should explain the experimental results.

**Example code for generating synthetic table.**

**/\* Make a table \*/**

**DROP TABLE** TEST;

**CREATE TABLE** TEST (a **INT**, b **INT**);

DELIMITER $$

**DROP PROCEDURE IF EXISTS** loopInsert $$

**CREATE PROCEDURE** loopInsert()

**BEGIN**

**DECLARE** i **INT** **DEFAULT** 1;

**WHILE** i <= **20000000** **DO**

**INSERT** **INTO** TEST (a, b) **VALUES** (i, i);

**SET** i = i + **1**;

**END** **WHILE**;

**COMMIT**;

**END**$$

DELIMITER ;

**SET** autocommit=**0**;

**CALL** loopInsert;

**COMMIT**;

**SET** autocommit=**1**;

**/\* Make a index \*/**

**ALTER** **TABLE** TEST **ADD** **INDEX**(a);

**/\* Compare the running time between index scan and full table scan at selectivity 50% \*/**

**SELECT** **SUM**(a)

**FROM** TEST

**WHERE** a > **10000000**;

**SELECT** **SUM**(b)

**FROM** TEST

**WHERE** b > **10000000**;

**(a) Comparison graph for running time over selectivity.**

**(b) Explain why index scan is faster or slower than full table scan depending on the selectivity in your comparison results.**

**As you can see in my graph, two lines are intersecting when the selectivity is about 93%.**

**When doing Index scan, first search all the target indices which is rowID, and then randomly access the row of content and find target in disk. However, Full table scan just access total contents sequentially in disk. So, if selectivity has very big value(93%), just searching sequentially one by one content can be faster than checking all indices because almost of contents in disk should be chosen and read, and the cost of random access used by Index scan become more expensive. But, in contrary, if selectivity go smaller than that(93%), the quantity of chosen contents go smaller. Then checking just indices before randomly read contents in disk will go faster.**