CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (FALL 2023)
PROF. ANDY PAVLO AND JIGNESH PATEL

Homework #2 (by Wiam Eddahri)

Due: **Sunday September 24 2023 @ 11:59pm**

IMPORTANT:

- Enter all of your answers into **Gradescope by 11:59pm on Sunday September 24 2023.**
- **Plagiarism**: Homework may be discussed with other students, but all homework is to be completed **individually**.

For your information:

- Graded out of **100** points; **3** questions total
- Rough time estimate: ≈4-6 hours (1-1.5 hours for each question)

Revision : 2023/09/15 10:33

Question	Points	Score
Storage Models	16	
Extendible Hashing	19	
B+Tree	65	
Total:	100	

Question 1: Storage Models ....[16 points]

Consider a database with a single table `T(andrew_id,full_name,graduation_year, department,gpa)`, where `andrew_id` is the *primary key*, and all attributes are the same fixed width. Suppose `T` has 5,000 tuples that fit into 500 pages, Ignore any additional storage overhead for the table (e.g., page headers, tuple headers). Additionally, you should make the following assumptions:

- The DBMS does *not* have any additional meta-data (e.g., sort order, zone maps).
- `T` does *not* have any indexes (including for primary key `andrew_id`)
- None of `T`'s pages are already in the buffer pool.
- Content-wise, the tuples of `T` will make each query run the longest possible (this assumption is critical for solving part **(a)**)
- The tuples of `T` can be in any order (this assumption is critical for solving part **(b)** when you compute the *minimum* versus *maximum* number of pages that the DBMS will potentially have to read)

(a) Consider the following query:

```
SELECT MAX(gpa) FROM T
    WHERE graduation_year == 2025;
```

i. **[4 points]** Suppose the DBMS uses the decomposition storage model (DSM) with implicit offsets. How many pages will the DBMS potentially have to read from disk to answer this query? (Keep in mind our assumption about the contents of `T`!)
□ 1-100 □ 101-200 □ 201-300 □ 301-500 □ ≥ 501 □ Not possible to determine

ii. **[4 points]** Suppose the DBMS uses the N-ary storage model (NSM). How many pages will the DBMS potentially have to read from disk to answer this query? (Keep in mind our assumption about the contents of `T`!)
□ 1-200 □ 201-300 □ 301-400 □ 401-500 □ ≥ 501 □ Not possible to determine

(b) Now consider the following query:

```
SELECT andrew_id, gpa FROM T
    WHERE gpa = (SELECT MAX(gpa) FROM T);
```

i. Suppose the DBMS uses the decomposition storage model (DSM) with implicit offsets.

α) **[4 points]** What is the *minimum* number of pages that the DBMS will potentially have to read from disk to answer this query?
□ 1 □ 2-5 □ 100-200 □ 201-299 □ ≥ 300 □ Not possible to

determine

β) **[4 points]** What is the *maximum* number of pages that the DBMS will potentially have to read from disk to answer this query?
□ 1 □ 2-5 □ 100-200 □ 201-299 □ ≥ 300 □ Not possible to determine

Question 2: Extendible Hashing................................[19 points]

Consider an extendible hashing structure such that:

- Each bucket can hold up to four records.
- The hashing function uses the lowest g bits, where g is the global depth.
- A new extendible hashing structure is initialized with $g = 0$ and one empty bucket

(a) Starting from an empty table, insert keys 0, 1, 4, 5.

i. **[2 points]** What is the global depth of the resulting table?
□ 0 □ 1 □ 2 □ 3 □ 4 □ None of the above

ii. **[2 points]** What is the local depth of the bucket containing 5?
□ 0 □ 1 □ 2 □ 3 □ 4 □ None of the above

(b) Starting from the result in **(a)**, you insert keys 10, 11, 12, 13.

i. **[2 points]** What is the global depth of the resulting table?
□ 0 □ 1 □ 2 □ 3 □ 4 □ None of the above

ii. **[2 points]** What is the local depth of the bucket containing 11?
□ 0 □ 1 □ 2 □ 3 □ 4 □ None of the above

(c) Starting from the result in **(c)**, you insert keys 2 and 14.

i. **[2 points]** What is the global depth of the resulting table?
□ 0 □ 1 □ 2 □ 3 □ 4 □ None of the above

ii. **[3 points]** Starting from the result in **(c)**, you insert keys 15, 3 and 7, what is the global depth?
□ 0 □ 1 □ 2 □ 3 □ 4 □ None of the above

iii. **[3 points]** Which value, if inserted, will hash to the same bucket as the bucket containing key 1?
□ 3 □ 7 □ 11 □ All of the above □ None of the above

(d) **[3 points]** Starting from the result in **(c)**, which **key(s)**, if inserted next, will cause a split that doubles the table's size?
□ 33 □ 64 □ 29 □ 61 □ All of the above □ None of the above

Question 3: B+Tree . . . . [65 points]

Consider the following B+tree.

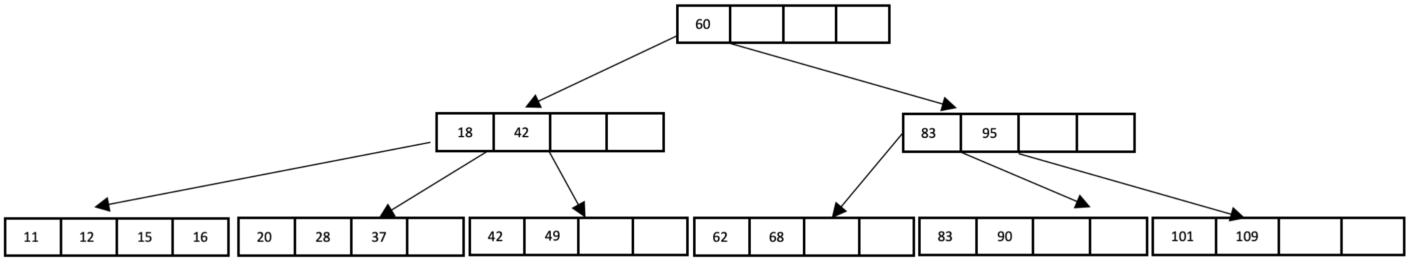


Figure 1: B+ Tree of order $d = 5$ and height $h = 3$.

When answering the following questions, be sure to follow the procedures described in class and in your textbook. You can make the following assumptions:

- A left pointer in an internal node guides towards keys $<$ than its corresponding key, while a right pointer guides towards keys $\geq$.
- A leaf node underflows when the number of **keys** goes below $\lceil \frac{d-1}{2} \rceil$.
- An internal node underflows when the number of **pointers** goes below $\lceil \frac{d}{2} \rceil$.

- You should always consider redistribution before trying to merge two nodes.

Note that B+ tree diagrams for this problem omit leaf pointers for convenience. The leaves of actual B+ trees are linked together via pointers, forming a singly linked list allowing for quick traversal through all keys.

Question 3 continues…

(a) **[5 points]** How many tree nodes must be fetched to answer the following query: Get all records with search key smaller than 42.

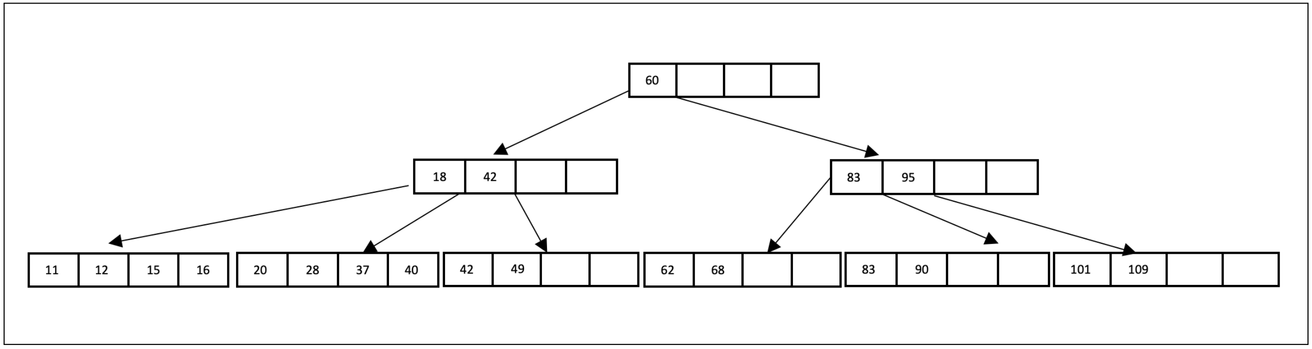
□ 6 □ 7 □ 5

□ 4 □ None

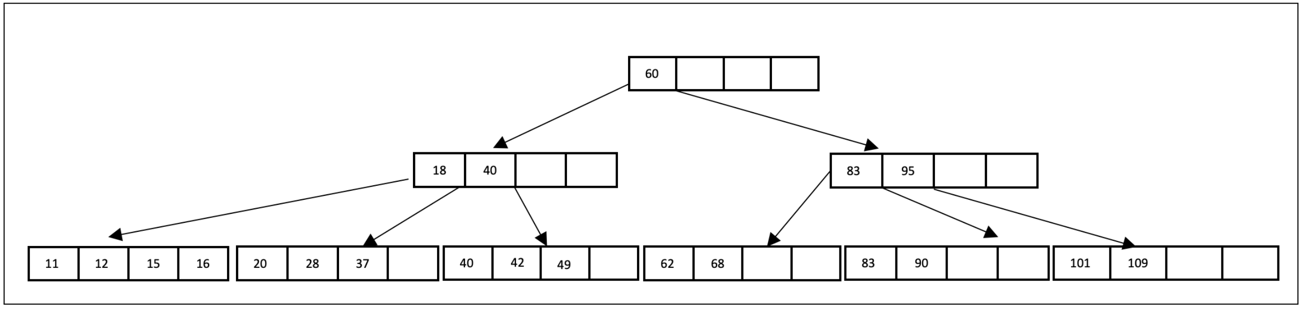
Question 3 continues. . .

(b) **[5 points]** Insert 40* into the B+tree. Select the resulting tree.

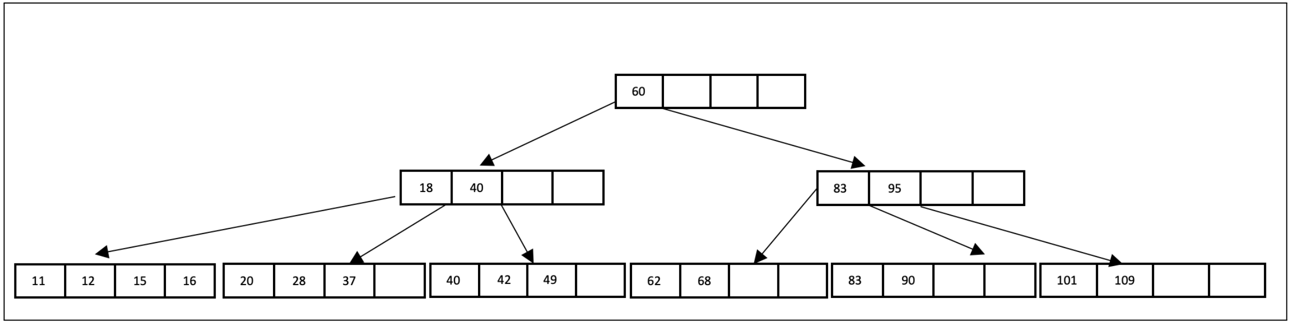
□ A)



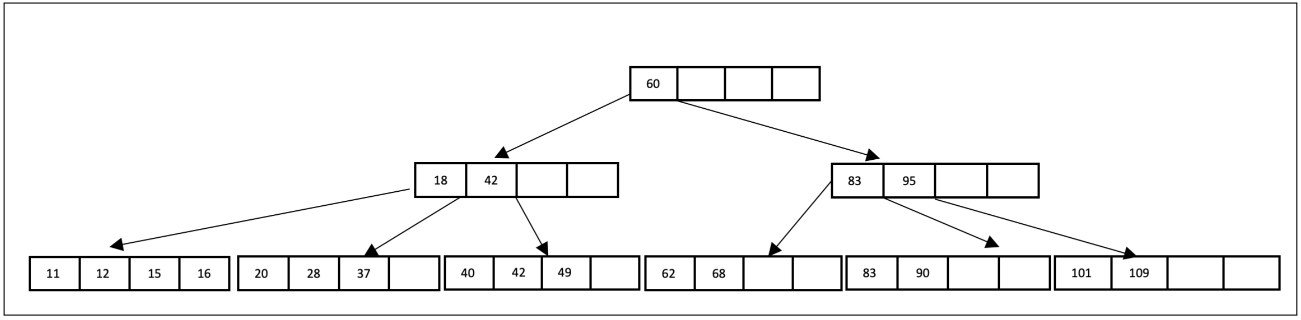
□ B)



□ C)

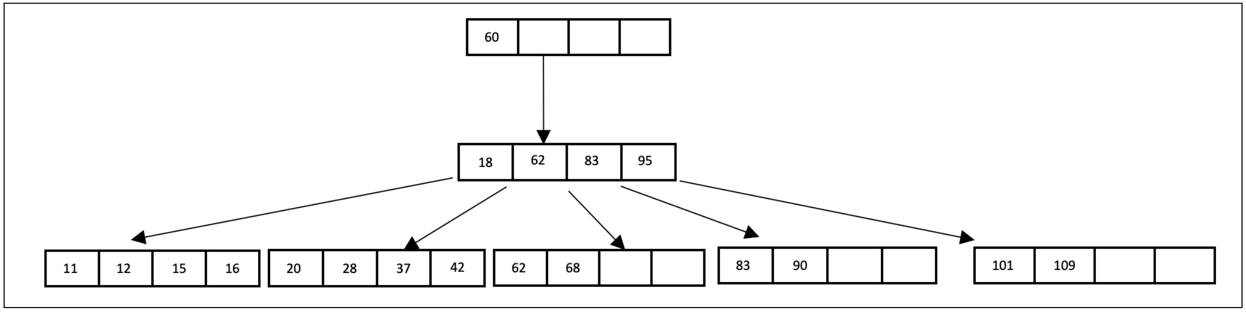


□ D)

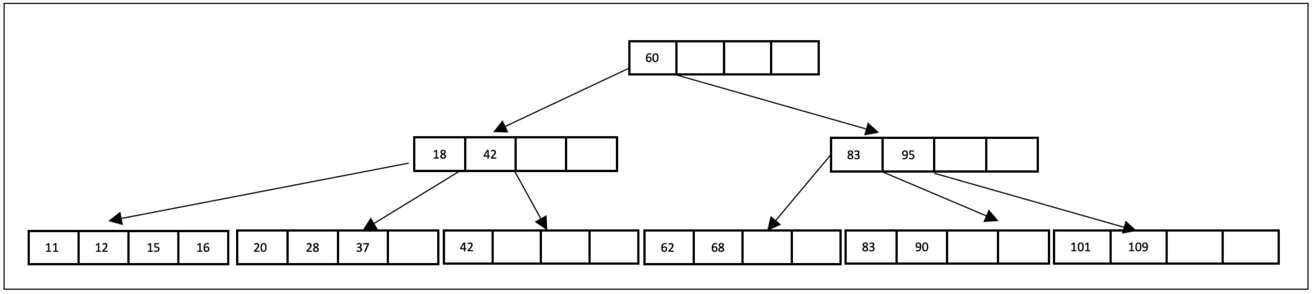


(c) **[5 points]** Starting with the intial tree, delete 49^*. Select the resulting tree.

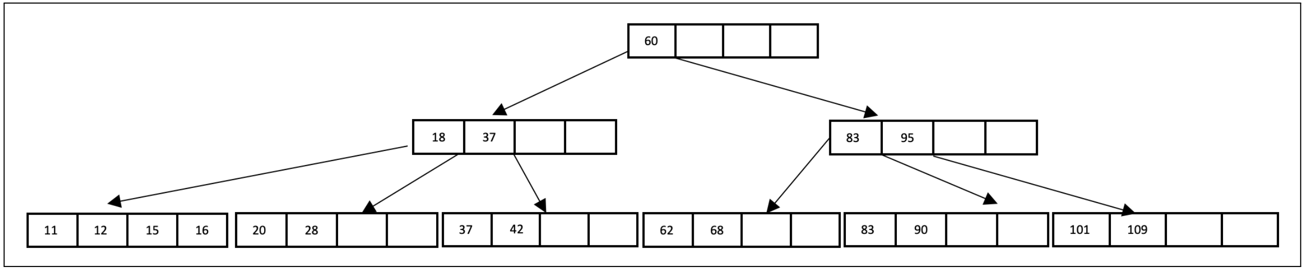
□ A)



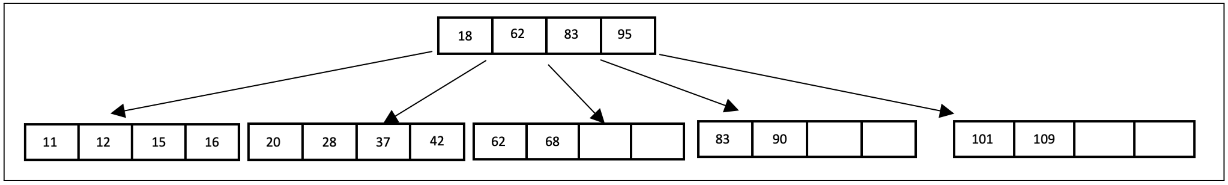
□ B)



□ C)

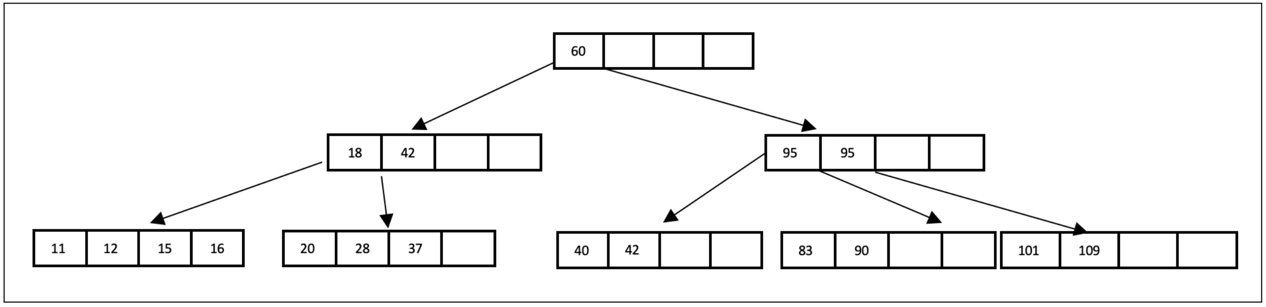


□ D)

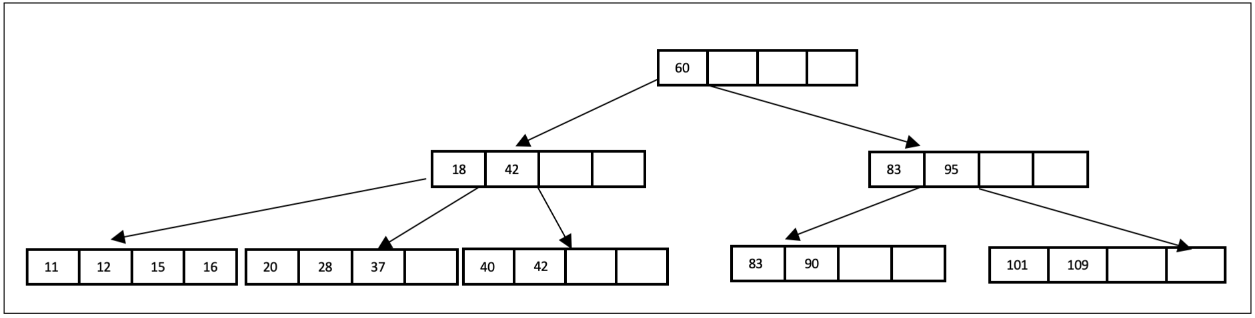


(d) **[5 points]** Starting with the intial tree, delete 62^* and delete 68^* . Select the resulting tree.

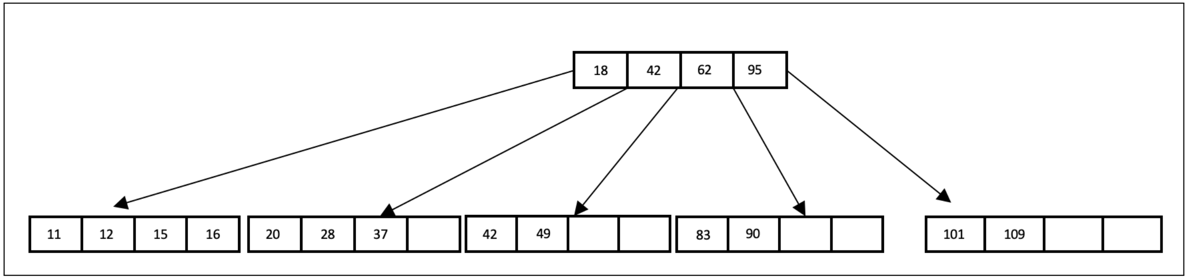
□ A)



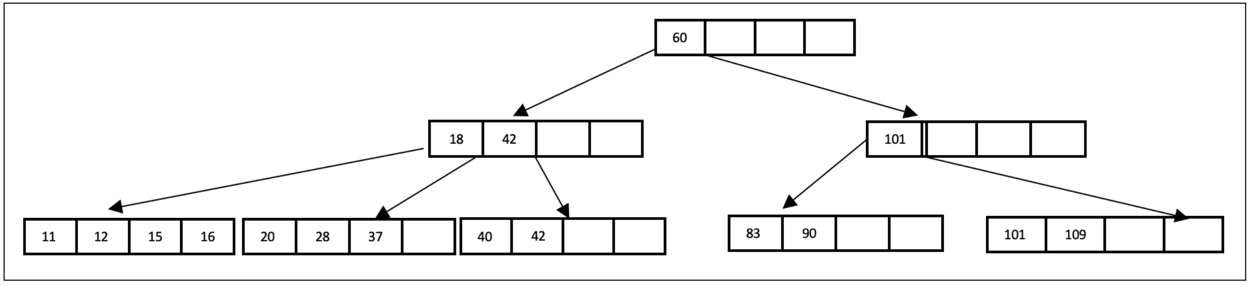
□ B)



□ C)



□ D)



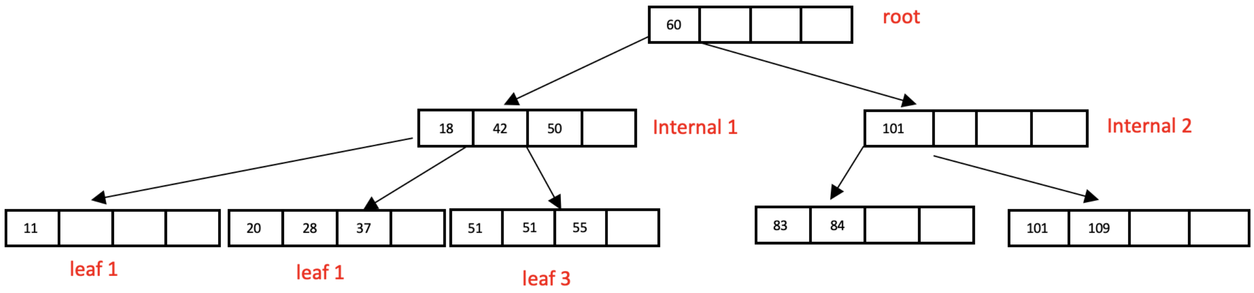


Figure 2: B+tree with violations

(e) The B+Tree shown in Figure 2 is invalid. That is, its nodes violate the correctness properties of B+Trees that we discussed in class. If the tree is invalid, select all the properties that are violated for each node. If the node is valid, then select 'None'. There will be **no** partial credit for missing violations.

Note:

- *If a node's subtrees are not the same height, the balance property is violated at that node only.*
- *If a node's subtrees contain values not in the range specified by the node's separator keys, the separator keys property is violated at that node.*

i. **[2 points]** Which properties are violated by **Leaf 3**?
□ Key order property □ Half-full property □ Balance property
□ Separator keys □ None

ii. **[2 points]** Which properties are violated by **Leaf 1**?
□ Key order property □ Half-full property □ Balance property
□ Separator keys □ None

iii. **[2 points]** Which properties are violated by **Internal Node 1**?
□ Key order property □ Half-full property □ Balance property
□ Separator keys □ None

iv. **[2 points]** Which properties are violated by **Internal Node 2**?
□ Key order property □ Half-full property □ Balance property
□ Separator keys □ None

v. **[2 points]** Which properties are violated by **Root**?
□ Key order property □ Half-full property □ Balance property
□ Separator keys □ None

(f) i. **[5 points]** A DBMS may potentially use separate buffer pools for a B+Tree's inner node pages and for its leaf node pages.
□ True □ False

ii. **[5 points]** A read-only thread needs to hold at most two latches at the same time.
□ True □ False

Question 3 continues. . .

iii. **[5 points]** A write thread needs to hold at most three write latches at the same time
□ True □ False

iv. **[5 points]** A write thread might hold only one write latch (Consider a tree that has more than one leaf node)
□ True □ False

v. **[5 points]** A read thread must release its latches in the order they were acquired (i.e., FIFO) to prevent concurrency errors.
□ True □ False

vi. **[5 points]** A write thread must release its latches in the order they were acquired (i.e., FIFO) to prevent concurrency errors.
□ True □ False

vii. **[5 points]** The minimum space utilization for a B+ tree index is 50 percent.
□ True □ False