Name	CWID

Exam
1

Wednesday, March 31, 2021
(3:30-6:00pm)

CS525 - Spring 2021 - MidTerm

Please leave this empty! 1.1 ☐ 2.1 ☐ 2.2 ☐ Sum ☐

Instructions

- **You have to hand in the assignment via blackboard**
- **This is an individual and not a group assignment. Fraud will result in 0 points**
- For your convenience the number of points for each part and questions are shown in parenthesis.
- There are 2 parts in this exam (50 points total)
  1. `Disk Organization` (5)
  2. `Index Structures` (45)

```
BY SUBMITTING THIS EXAM THROUGH THE ONLINE SYSTEM, I AFFIRM ON MY HONOR THAT I AM AWARE OF
THE STUDENT DISCIPLINARY CODE, AND (I) HAVE NOT GIVEN NOR RECEIVED ANY UNAUTHORIZED AID TO/FROM
ANY PERSON OR PERSONS, AND (II) HAVE NOT USED ANY UNAUTHORIZED MATERIALS IN COMPLETING MY
ANSWERS TO THIS TAKE-HOME EXAMINATION.
```

Part 1 Disk Organization (Total: 6 Points)

Question 1.1 Record Layout (5 Points)

Suppose blocks of 2048 bytes are used to store variable-length records. The fixed part of the block header is 24 bytes, and includes a count of the number of records. In addition, the header contains a 8-byte pointer (offset) to each record in the block. The first record is placed at the end of the block, the second record starts where the first one ends, and so on. The size of each record is not stored explicitly, since the size can be computed from the pointers.

1. How many records of size 100 bytes could we fully fit (no spanning) in such a block? (The block is designed to hold variable size records, but these records by coincidence happen to be all the same size.)

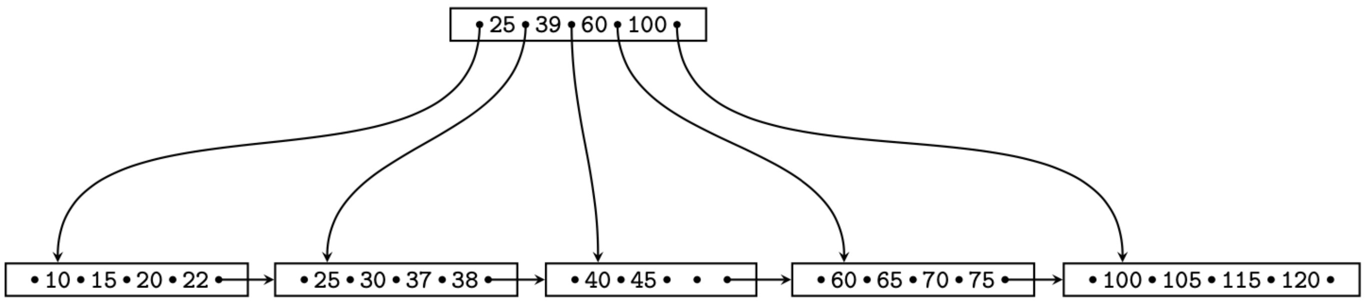
2. Suppose we want to fully store 10 records in the block, again all of the same size. What is the maximum size of such records?

3. Now assume the block layout is changed to accommodate only fixed size records. In this case, the block header is still 24 bytes and contains the common length of the records in the block. How many records of size 200 bytes could we fully fit in such a block?

Part 2 Index Structures (Total: 45 Points)

Question 2.1 B$^+$ Index Structures (25 Points)

Consider the following `B+-tree` ($n = 4$).



a) For the following query, assume each record in the answer resides in a different page. Calculate the number of disk I/Os if we use this index to answer the following query. Briefly explain your calculation.

```
SELECT * FROM R WHERE grade ≥ 15 AND grade ≤ 65;
```

b) Draw the modified tree after insert 125, delete 40, insert 35, and then delete 60. You need to show the resulted tree after performing each operation.

When splitting or merging nodes follow these conventions:

- **Leaf Split**: In case a leaf node needs to be split, the left node should get the extra key if the keys cannot be split evenly.
- **Non-Leaf Split**: In case a non-leaf node is split evenly, the "middle" value should be taken from the right node.
- **Node Underflow**: In case of a node underflow you should first try to redistribute and only if this fails merge. Both approaches should prefer the left sibling.

Scratch Page - This page is intentionally left blank

Scratch Page - This page is intentionally left blank

Scratch Page - This page is intentionally left blank

Question 2.2 Extendible Hash Table (20 Points)

Consider indexing the following key values using an `Extendible Hash Table`. Their corresponding hash value `h(x)` is already computed (see Table below). Assume each bucket can hold at most 2 data items.

x	h(x)
A	1011
B	1101
C	1111
D	0101
E	1110
F	1110
G	0010

Suppose the keys are to be inserted in ascending alphabetical order, i.e., `A, B, C,..., G`. The hash table is initially empty. Draw the hash table after all the keys are inserted. Show the keys themselves in the buckets, not the hash value. Be sure to indicate i, the number of bits in the hash value that are used. For each bucket, also indicate the number of hash function bits that are used for that bucket.

Scratch Page - This page is intentionally left blank

Scratch Page - This page is intentionally left blank

Scratch Page - This page is intentionally left blank