CMPT 454 Fall 2017
Assignment 2

Instructor: Juhua Hu
TA: Zahra Zohrevand

Total points: 102 (10% of final grade)
Due date: Midnight of Tuesday, October 17th, 2017

1 Data Entries: Alternatives [24 points]

Consider the instance of the Students relation shown in Table 1, sorted by *age*: For the purposes of this question, assume that these tuples are stored in a sorted file in the order shown; the fist tuple is on page 1, the second tuple is also on page 1; and so on. Each page can store up to three data records; so the fourth tuple is on page 2.

Table 1: An instance of the Students relation, sorted by *age*

sid	*name*	*login*	*age*	*gpa*
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	19	3.2
53650	Smith	smith@math	19	3.8

Show what the data entries in each of the following indexes contain. If such an index cannot be constructed, say so and explain why. For the definition of primary or secondary index, please stick to our lecture notes.

1. A secondary dense index on *age* using Alternative (1). [3 points]
2. A primary dense index on *age* using Alternative (2). [3 points]
3. A primary sparse index on *age* using Alternative (2). [3 points]
4. A primary sparse index on *age* using Alternative (3). [3 points]
5. A secondary dense index on *gpa* using Alternative (2). [3 points]
6. A secondary sparse index on *gpa* using Alternative (2). [3 points]

7. A primary dense index on *gpa* using Alternative (2). [3 points]

8. A secondary dense index on *name* using Alternative (3). [3 points]

2 B+ Tree Index [38 points]

For the following set of data entries

$$(2^*, 3^*, 5^*, 7^*, 11^*, 17^*, 19^*, 23^*, 29^*, 31^*)$$

1. Construct a B+ tree, whose order is 2. Assume that the tree is initially empty and entries are inserted in an ascending order. [17 points]

2. Based on the B+ tree constructed in question 1, show the B+ tree after we insert $9^*, 10^*, 8^*, 6^*, 34^*$ in this specific sequence. Within this step, what is the I/O cost to insert 9^* supposing that the root is in the main memory? [19 points]

3. Suppose we want to delete 2^* from the B+ tree of question 2, what is the total I/O cost (the root is in the main memory)? Explain the calculation process. [2 points]

3 Hash Index [32 points]

Consider an extendible hash index as shown in Table 2, where the hash values are presented by their binary representations (8 bits for each value), the *directory* has two entries and therefore uses just the last bit of the hash value. We assume each bucket can fit at most four data entries.

Table 2: An initial state of the extendible hash index (gd means global depth and ld means local depth)

gd=1	*bucket*	*ld*
0	0011 1110, 1101 0110, 0001 1010	1
1	1111 1101, 1011 0011, 0011 1001, 1110 1101	1

1. Show the index after inserting a data entry with hash value "0011 0011". [8 points]

2. Show the index after inserting "0101 0111", "1011 1100", "1001 0101", "0110 0001" based on the result of question 1. [13 points]

3. Consider a linear hash index with the initial state similar as Table 2, show the index after inserting a data entry with hash value "0011 0011". We assume that split is triggered by creating/inserting into an overflow page. [5 points]

4. Show the linear hash index after inserting "0101 0111" based on the result of question 3. We assume that split is triggered by creating/inserting into an overflow page. [6 points]

4 Query Optimization - Overview [8 points, 1 for each]

Consider the following schema with the Sailors relation:

$$Sailors(\underline{sid : integer}, sname : string, rating : integer, age : real)$$

For each of the following indexes, list whether the index matches the given selection conditions (match or no match). If there is no match, explain why.

1. A B+ tree index on the search key ⟨Sailors.sid, Sailors.age⟩.
   (a) $\sigma_{Sailors.sid<50{,}000 \wedge Sailors.age=21}(Sailors)$
   (b) $\sigma_{Sailors.age>21 \wedge Sailors.sid=50{,}000}(Sailors)$
   (c) $\sigma_{Sailors.sid=50{,}000}(Sailors)$
   (d) $\sigma_{Sailors.age=21}(Sailors)$

2. A hash index on the search key ⟨Sailors.sid, Sailors.age⟩.
   (a) $\sigma_{Sailors.sid=50{,}000 \wedge Sailors.age=21}(Sailors)$
   (b) $\sigma_{Sailors.age>21 \wedge Sailors.sid=50{,}000}(Sailors)$
   (c) $\sigma_{Sailors.sid=50{,}000}(Sailors)$
   (d) $\sigma_{Sailors.age=21}(Sailors)$