The assignment is to be turned in before Midnight (by 11:59pm) on January 31st, 2016. You should turn in the solutions to this assignment as a pdf file through the TEACH website. The solutions should be produced using editing software programs, such as LaTeX or Word, otherwise they will not be graded. Trees can be drawn on paper and scanned.

1: Operating System Support for Database Management System (0.5 points)

Describe a situation where LRU replacement is a bad strategy for buffer management.

Solution:

Situation 1: When there is sequential or random access to blocks which will not be rereferenced. LRU is a bad strategy because it keeps blocks in cache after being read, even though they are not going to be referenced again. LRU discards these blocks only when the cache is full and a new block must be brought to cache. If the DBMS knows that a block won’t be referenced again, it can toss it immediately.

Situation 2: When there is sequential access to blocks which will be cyclically rereferenced. LRU is a bad strategy because when the cache is full, it discards the least recently used block. Then, because blocks are cyclically rereferenced, the cache will always be full, and there will be a disk access in all next block accesses.

2: B+ Tree Indexing (1.5 points)

Consider the B+ tree index shown in Figure 1. Each intermediate node can hold up to five pointers and four key values. Each leaf can hold up to four pointers to data, and leaf nodes are doubly linked as usual, although these links are not shown in the figure. Note that A, B and C are subtrees of the B+ tree, but are not fully specified. Answer the following questions.

![Figure 1: Tree for question 2.](image)

1. Show the B+ tree that would result from inserting a record with search key 109 into the tree.

   Solution: The B+ tree is shown in Figure 2.

   ![Figure 2: Tree for question 2.1.](image)

2. Show the B+ tree that would result from deleting the record with search key 50 from the original tree.
Solution: The B+ tree is shown in Figure 3.

```
  10  20  30  80
   A   B   C
     |     |
     35   42
     |     |
     65   90
```

Figure 3: Tree for question 2.2.

3. Name a search key value such that inserting it into the (original) tree would cause an increase in the height of the tree.

Solution: Any search key in the range $[30..79]$.

4. What can you infer about the contents and the shape of A, B and C subtrees?

Solution: We can infer several things about subtrees A, B, and C. First of all, each of them is a leaf node, since their “siblings” are leaf nodes. Also, we know the ranges of these trees (assuming duplicates fit on the same leaf): subtree A holds search keys less than 10, B contains keys $\geq 10$ and $< 20$, and C has keys $\geq 20$ and $< 30$. In addition, each intermediate node has at least 2 key values and 3 pointers.

3: B+ Tree Indexing (1.5 points)

Suppose that a block can contain at most four data values and that all data values are integers. Using only B+ trees of degree 2, give examples of each of the following:

1. A B+ tree whose height changes from 2 to 3 when the value 42 is inserted. Show your structure before and after the insertion.

Solution: The tree before inserting value 42 is shown in Figure 4. The tree after inserting value 42 is shown in Figure 5.

```
  10  20  30  80
   2   6
     |
     10
     |
     13
     |
     16
```

Figure 4: Tree for question 3.1 (before).

```
  10  20
   2   6
     |
     10
     |
     13
     |
     16
```

Figure 5: Tree for question 3.1 (after).
2. A B+ tree in which the deletion of the value 42 leads to a redistribution. Show your structure before and after the deletion.

**Solution:** The tree before deleting value 42 is shown in Figure 6. The tree after deleting value 42 is shown in Figure 7.

```
2 6  
10 13 16  
25 42

Figure 6: Tree for question 3.2 (before).
```

```
2 6  
10 13  
16 25

Figure 7: Tree for question 3.2 (after).
```

4: B+ Tree Indexing (1.5 points)

Consider the instance of the Students relation shown in Figure 8.

1. To reduce the number of I/O access in index search, each B+ tree node should fit in a block. Let \( sid \) be an integer requiring 32 bits. Let a pointer require 64 bits. If the block size is 56 bytes (consisting of 8 bits), what should be the minimum degree of the B+ tree index on \( sid \)?

**Solution:**

\[
2d \times 32 + (2d + 1) \times 64 \leq 56 \times 8
\]
\[
64d + 128d + 64 \leq 448
\]
\[
192d \leq 384
\]
\[
d \leq 2
\]

Hence, the degree of the B+ tree must be between 1 (minimum) and 2 (maximum).

2. Show a B+ tree index on \( sid \) of degree calculated in part 1 for all records in Figure 8.

**Solution:** A B+ tree with degree 1 is shown in Figure 9.

A B+ tree with degree 2 is shown in Figure 10. Either of these B+ trees are acceptable for the solution. B+ trees that are created using other orders of the input tuples are also acceptable.
Figure 8: An Instance of the Students Relation.

Figure 9: Tree for question 4.2 with degree 1.

Figure 10: Tree for question 4.2 with degree 2.