HOMEWORK #6

Problem 1

Please highlight the changes (or only describe/show the changes) to make it easier for the grader to check your work. For example, if you're splitting node F, you could say split node F into F and F'. Node F has keys ... and F' has keys ...

Consider the following B+ tree. The structure is described first, followed by the key values in each node.

n = 8. (Therefore internal nodes other than the root must have between 4 and 8 children and leaves must have between 4 and 7 key values.)

Structure:

```
The root is node A.
Node R has two children: A and B.
A has four children: C, D, E, F.
B has eight children: G, H, I, J, K, L, M, N.

Key values:
R: 40
A: 10, 18, 32
B: 48, 62, 74, 82, 100, 200, 300
C:  2,  4,  6,  8
D: 10, 12, 14, 16
E: 18, 20, 22, 24, 26, 28, 30
F: 32, 34, 36, 38
G: 40, 42, 44, 46
H: 48, 50, 52, 54, 56, 58, 60
I: 62, 64, 66, 68, 70, 72
J: 74, 76, 78, 80
K: 82, 84, 86, 88
L, M, N: details not needed for this problem
```

1. What nodes are visited and, in each what key comparisons are made, when doing each of the following searches:
   (a) find record with key value 58
   (b) find record with key value 32
   (c) find all records with key values between 24 and 42,assuming the B+ tree is a secondary index on the data file.

(d) find all records with key values between 24 and 42,assuming the B+ tree is a primary index on the data file.

2. Describe the tree after each of the following operations. DO EACH OF THESE SEPARATELY, STARTING WITH THE ORIGINAL TREE DESCRIBED ABOVE:

   (a) insert 11

   (b) insert 19

   (c) insert 59

Problem 2

Consider a B+ tree with root, two additional levels of internal nodes, and leaf nodes (4 nodes on paths from root down to and including leaf) that is a dense index on some attribute a of relation r. Assume a is UNIQUE, i.e. no two tuples of r have the same value of attribute a. Suppose $n = 15$ and each internal node, including the root, has 10 children and assume each leaf node has 10 values. Assume each node is on its own disk block. Assume each data block in r has 4 tuples (with different values of a).

1. How many different values of a does r have?
2. What is the numerical value of b_r (the number of data blocks r occupies)?
3. Consider execution of the query `SELECT * FROM r WHERE a = 'foo'` How long will it take in the worst case to execute the query using a sequential scan of the whole file (in terms of the block access (seek + rotational latency) time, t_s, and the block transfer time, t_T) ?
4. How long will it take to execute the query using the index (in terms of t_s and t_T) ?

Problem 3

Consider the following pseudo-code for an application that executes some queries (using the university database)

```
q1 = `` SELECT id FROM TAKES WHERE course_id = 'CS-3083' AND grade = 'A' ''
rs = result of executing q1
for each tuple t in rs
     {
           cur_id = t[id]   # bind the id value to program variable cur_id
           q2 = `` SELECT name FROM student WHERE id = ? ''
           rs2 = result returned by binding ? to cur_id and execute q2
           pass the names in rs2 to a function that formats output
     }
```

For each row of the result of q1, this code binds the id from that row into q2 then executes q2. The overall result is the names of students who got an 'A' in 'CS-3083'

1. Give a formula for the number of block transfers and a formula for the number of seeks needed to execute this pseudocode, in terms of:

- b_t = the number of blocks occupied by the takes relation
- b_s = the number of blocks occupied by the student relation
- n_A = the number of students who got an 'A' in 'CS-3083'

2. Write a single query involving a join that finds the names of students who got an 'A' in 'CS-3083'. Give a formula for the number of block transfers and a formula for the number of seeks needed to execute this query for some join algorithm (state which algorithm you're using and define any additional parameters you need.)

3. Comment on what your answers to (1) and (2) imply about the nested loop application style used above, in which a query is instantiated and executed multiple times within a loop.