1. Suppose that we have an ordered file with 30,000 records stored on a disk with block size 1024 bytes. File records are of fixed size with record length 100 bytes.

a) Find the number of block accesses required to search for a record using a binary search.

Number of records per block = $\lfloor(1024/100)\rfloor$ = 10
Number of blocks needed for the file = $\lceil(30000/10)\rceil$ =3000
Number of block accesses needed to search for a record using a binary search = $\lceil\log_2 3000\rceil$ = 12

b) Suppose that the search key field of the file is 9 bytes long, a pointer is 6 bytes long, and a primary index is constructed for the file with one index entry per data block. Find the number of block accesses required to search for a record using the index.

Number of index entries per block = $\lfloor(1024 / (9+6))\rfloor$ = 68
Number of index entries = number of blocks in the data file = 3000
Number of index blocks = $\lceil(3000/68)\rceil$ = 45
Number of block accesses needed to search for an index record using a binary search = $\lceil\log_2 45\rceil$ = 6
Number of block accesses needed to search for a record using the index = 6 + 1 = 7

c) How many levels are required to construct a multilevel index on the primary index in b) such that there is only one index block at the top level? Find the number of block accesses required to search for a record using the multi-level index.

From b), the number of first-level index blocks = 45
Number of second-level index entries = number of first-level blocks = 45
Number of second-level index blocks = $\lceil(45/68)\rceil$ = 1

Since the second level has only one block, it is the top index level. Hence, the number of levels required is 2.
The number of block accesses to search for a record = 2 + 1 = 3

2. Consider a disk with block size 512 bytes. Suppose that the search key field of a file is 9 bytes long and a pointer is 6 bytes long. We want to construct a B^+-tree index for the file and a node of the B^+-tree is made to be the same size as a disk block.

a) What is the largest integer value of *n* for the B^+-tree?

For a B^+-tree of order *n*, the following inequality must be satisfied.
$(n \times 6) + ((n-1) \times 9) \leq 512$
$15 \times n \leq 521$
So, $n = 34$

b) What are the largest and the least number of search key values that can be stored at the leaf level of a 4-level B^+-tree?

Root: 1 node, 34 children
2nd level: 34 nodes, 1,156 children
3rd level: 1,156 nodes, 39,304 children
Leaf level: 39,304 nodes, 1,297,032 search key values

Root: 1 node, 2 children
2nd level: 2 nodes, 34 children
3rd level: 34 nodes, 578 children
Leaf level: 578 nodes, 9,826 search key values

3. Consider a B^+-tree and a given function `find(V)`, which returns leaf node *C* and index *i* such that $C.P_i$ points to the record with search key value *V*, if such a record exists. Write a pseudocode for a procedure `printRange(L, U)` to find and print all records with search key values in a specified range (*L*, *U*), assuming both *L* and *U* exist in the tree and the number of keys in a leaf node is known. Such queries are called ***range queries***.

procedure printRange(*value L*, *U*)
/* assume that both *L* and *U* exist in the tree */
    Set done = false;
    Set (*C*, *i*) = find(*L*);
    repeat
        while ($i \leq$ number of keys in *C* **and** $C.K_i \leq U$)
            Print record pointed to by $C.P_i$
            Set $i = i+1$
        if ($i >$ number of keys in *C*)
            then ($C=C.P_n$; $i = 1$)
            else Set done=true;
    until (done **or** *C* is null)