Task 1 (5 marks)

Consider a relational table `EMPLOYEE(e#, name, salary, position)` where an attribute `e#` is a primary key.

Assume that:

(i) a relational table `EMPLOYEE` occupies 10^2 data blocks,

(ii) a relational table `EMPLOYEE` contains 10^3 rows,

(iii) an attribute name has `800` distinct values,

(iv) an attribute salary has `20` distinct values,

(v) an attribute position has `50` distinct values,

(vi) a primary key is automatically indexed,

(vii) the attributes `salary` and `position` are indexed,

(viii) all indexes are implemented as B*-trees with a fanout equal to `10`,

(ix) a leaf level of an index on attribute `salary` consists of `5` data blocks,

(x) a leaf level of an index on attribute `position` consists of `20` data blocks.

Find each of the following queries describe how a database system plans to compute the queries, (i.e. provide detailed query execution plans) and determine the total number of read block operations needed to compute the query. Show ALL computations performed to get the final answer.

(1)

```
SELECT DISTINCT salary
FROM EMPLOYEE;
```

Solution

An attribute `salary` is indexed. Therefore the system will horizontally traverse a leaf level of B*-Tree that implements and index on an attribute `salary`.

Total number of blocks read is equal to the total number of blocks at leaf level of an index on attribute `salary`.

Total number of blocks read = `5`

(2)

```
SELECT position, COUNT(*)
FROM EMPLOYEE
GROUP BY position;
```

Solution

An attribute `position` is indexed. Therefore the system will horizontally traverse a leaf level of B*-Tree that implements an index on an attribute `position` and it will count the total number of row identifiers associated with each index key encountered at leaf level of B*-Tree.

Total number of blocks read is equal to the total number of blocks at leaf level of an index on attribute salary.

Total number of blocks read = 20

(3)

```
SELECT *
FROM EMPLOYEE
WHERE e# = 007 AND position = 'boss';
```

Solution

A primary key e# is indexed. The system will vertically traverse B*-Tree that implements an index on an attribute e#. When a key 007 is found the system will take a row identifier associated with the key and it will read 1 data block from an implementation of a relational table EMPLOYEE.

Total number of blocks read = total number of blocks read to vertically traverse the index + 1 = $(\log_{10}10^3 +1) +1 = 5$

(4)

```
SELECT *
FROM EMPLOYEE
WHERE position = 'boss';
```

Solution

An attribute position is indexed. Therefore the system will vertically traverse a B*-Tree that implements an index on an attribute salary. When a key 'boss' is found the system will take the row identifiers associated with the key and it will read data blocks from an implementation of a relational table EMPLOYEE.

Blocking factor = $10^3/10^2 = 10$

Total number of blocks read = total number of blocks read to vertically traverse the index + total number of blocks read from an implementation of a relational table EMPLOYEE = $(\log_{10}50 +1) + (10^3/50 + 10^3/(50*10))/2$

(5)

```
SELECT *
FROM EMPLOYEE
WHERE position = 'boss ' AND salary = 1000;
```

Solution

Both attributes position and salary are indexed. Therefore the system will vertically traverse a B*-Tree that implements an index on an attribute position to find a set of row identifiers associated with a key 'boss'. Then, the system will vertically traverse a B*-Tree that implements an index on an attribute salary to find a set of row identifiers associated with a key 1000. Next, the system will compute an intersection of the sets of row identifiers found in the previous two steps. Finally, the system will use a set of row identifiers obtained from the intersection and it will read data blocks from an implementation of a relational table `EMPLOYEE`.

Blocking factor = $10^3/10^2 = 10$

Total number of blocks read = total number of blocks read to vertically traverse an index on attribute `position` + total number of blocks read to vertically traverse the index on attribute `salary` + total number of blocks read to compute an intersection of the sets of row identifiers obtained from two previous steps + total number of blocks read to read the blocks from an implementation of a relational table `EMPLOYEE` =

$(\log_{10}50 + 1) + (\log_{10}20 + 1) + 0 +$
$(10^3/(50*20) + 10^3/(50*20*10))/2$

(6)

```
SELECT MAX(SALARY)
FROM EMPLOYEE;
```

Solution

An attribute `salary` is indexed. Therefore the system will access a leaf level of B*-Tree that implements and index on an attribute `salary` from a side of the largest values of the key.

Total number of blocks read is equal to the total number of blocks read to find the maximum value of a key at leaf level of an index on attribute `salary`.

Total number of blocks read = 1

(7)

```
SELECT *
FROM EMPLOYEE
WHERE position = 'boss ' OR salary = 1000;
```

Solution

Both attributes `position` and `salary` are indexed. Therefore the system will vertically traverse a B*-Tree that implements an index on an attribute `position` to find a set of row identifiers associated with a key `'boss'`. Then, the system will vertically traverse a B*-Tree that implements an index on an attribute `salary` to find a set of row identifiers associated with a key `1000`. Next, the system will compute an union of the sets of row identifiers found in the previous two steps. Finally, the system will use a set of row identifiers obtained from the union and it will read data blocks from an implementation of a relational table `EMPLOYEE`.

Blocking factor = $10^3/10^2 = 10$

Total number of blocks read = total number of blocks read to vertically traverse an index on attribute `position` + total number of blocks read to vertically traverse the index on attribute `salary` + total number of blocks read to compute a union of the sets of row identifiers obtained from two previous steps + total number of blocks read to read the blocks from an implementation of a relational table EMPLOYEE =

$(\log_{10}50 + 1) + (\log_{10}20 + 1) + 0 +$
$((10^3/50 + 10^3/20 - 10^3/(50*20)) +$
$(10^3/50 + 10^3/20 - 10^3/(50*20))/10))/2$

(8)

```
SELECT salary
FROM EMPLOYEE;
```

Solution

An attribute salary is indexed. Therefore the system will horizontally traverse a leaf level of B*-Tree that implements and index on an attribute salary.

Total number of blocks read is equal to the total number of blocks at leaf level of an index on attribute salary.

Total number of blocks read = 5

(9)

```
SELECT salary, position
FROM EMPLOYEE;
```

Solution

The system will traverse entire relational table EMPLOYEE.

Total number of blocks read is equal to the total number of blocks required for an implementation of a relational table EMPLOYEE.

Total number of blocks read = 10^2

(10)

```
SELECT *
FROM EMPLOYEE
ORDER BY salary DESC;
```

Solution

An attribute salary is indexed. Therefore, the system will horizontally traverse a leaf level of B*-Tree that implements and index on an attribute salary starting from a side of the largest value of index key (it will implement sorting on an attribute salary). Then, with each key accessed at leaf level of the index, the system will use the row identifiers associated with the key to access a relational table EMPLOYEE.

Total number of blocks read is equal to the total number of blocks at leaf level of an index on attribute salary + total number of blocks required for implementation of a relational table EMPLOYEE.

Total number of blocks read = $5 + 10^2$

End of sample solution