The following tables will be used for some questions in this challenge:

```
CREATE TABLE Employee ( EmpNo INT, EmpName VARCHAR, Salary INT, DeptNo INT, PK EmpNo, FK DeptNo Ref. Dept);
CREATE TABLE Dept ( DeptNo INT, DeptName VARCHAR );
CREATE TABLE EmpBonus ( BonusNo INT, EmpNo INT, BonusAmt INT, Year INT, PK BonusNo, FK EmpNo Ref. Employee );
```

**Question 1**

B-Tree is supported in almost every enterprise level RDBMS in order to enhance the speed of searches. What is the biggest factor in determining the number of physical record accesses in a given key search?

A) Number of keys in a node  
B) Tree height  
C) Sum of the leaf nodes  
D) Datatype of node data (i.e. char vs. int)

**Question 2**

Let’s assume that the Employee table produces a full binary tree. If the height of this tree is 5, what is the maximum number of nodes contained in the binary tree?

A) 48  
B) 55  
C) 31  
D) 63

**Question 3**

```
SELECT * FROM Employee WHERE Employee.EmpNo = '1';
```

Does eliminating implicit type conversion increase query performance of the above query? If so, why?

A) SQL won’t have to convert the value ’1’ to 1 one time  
B) SQL won’t have to convert the value ’1’ to 1 for every comparison  
C) SQL can use bitmap indexing if there’s no conversion  
D) This will not increase query performance, and can be left as it is
Question 4
Implicit type conversions, extra join operations, and table usage in join conditions can all lead to slower query execution. Taking into the account the tables created above, what should be changed in the following SQL SELECT statement to improve performance?:

```
FROM Employee, Emp-Bonus, Dept
WHERE EmpBonus.BonusAmt > '10,000' AND EmpBonus.EmpNo BETWEEN 1 AND 100 AND Employee.DeptNo = Dept.DeptNo AND Employee.EmpNo = EmpBonus.EmpNo;
```

A) Change: '10,000' to 10,000
B) Change: EmpBonus.EmpNo BETWEEN ... to Employee.EmpNo BETWEEN ...
C) Eliminate: Employee.DeptNo = Dept.DeptNo
D) All of the above.

Question 5 In an example scenario, most of the queries made on the tables described above look similar to the following example. Would denormalizing the tables Dept and Employee help improve query performance?

```
SELECT Dept.DeptName, Employee.EmpName FROM Dept, Employee
WHERE Dept.DeptNo = Employee.DeptNo;
```

A) Denormalization would not affect this specific query
B) Denormalization will improve the query’s performance because there will no longer need to be a join between Dept and Employee
C) Denormalization will decrease the performance of this query because the denormalized table will break the 3NF/BCNF form of the database
Answers

Question 1: B

Question 2: D

Question 3: C

Question 4: D, all of the above. Breakdown:
A is correct because EmpBonus.BonusAmt is an INT, and '10,000' is a CHAR, so there is an implicit conversion from a CHAR to an INT. This implicit conversion eliminates SQL's ability to do an index search.

B is correct because there is a 1-M relationship between Employee as the parent and EmpBonus as the child, so the parent should be used for the condition on the joining column to reduce the number of rows in the parent table.

C is correct because Dept is not needed in this query, so eliminating it reduces the number of joins.

Question 5: B