Day-35 MCQ

Question **1**

Correct

Mark 1.00 out of 1.00

What is the worst case time complexity for search, insert and delete operations in a general Binary Search Tree?

Select one or more:

☑ a. O(n) for all
✓

Explanation: In skewed Binary Search Tree (BST), all three operations can take O(n). See the following example BST and operations.

```
            10
          /
       20
      /
     30
    /
   40

Search 40.
Delete 40
Insert 50.
```

□ b. O(Logn) for all

□ c. O(Logn) for search and insert, and O(n) for delete

□ d. O(Logn) for search, and O(n) for insert and delete

Your answer is correct.

The correct answer is: O(n) for all

Question **2**

Correct

Mark 1.00 out of 1.00

What is the maximum height of any AVL-tree with 7 nodes? Assume that the height of a tree with a single node is 0.

Select one or more:

□ a. 2

☑ b. 3
✓

Explanation: AVL trees are binary trees with the following restrictions.
1) the height difference of the children is at most 1.
2) both children are AVL trees
Following is the most unbalanced AVL tree that we can get with 7 nodes

```
                   a
                /   \
              /       \
             b         c
           /  \       /
          /    \     /
         d      e   g
        /
       /
      h
```

□ c. 4

□ d. 5

Your answer is correct.

The correct answer is: 3

Question **3**

Correct

Mark 1.00 out of 1.00

Recursion is a method in which the solution of a problem depends on ____________

Select one or more:

□ a. Larger instances of different problems

□ b. Larger instances of the same problem

☑ c. Smaller instances of the same problem ✓

Explanation: In recursion, the solution of a problem depends on the solution of smaller instances of the same problem.

□ d. Smaller instances of different problems

Your answer is correct.

The correct answer is: Smaller instances of the same problem

Question **4**

Correct

Mark 1.00 out of 1.00

How many times is the recursive function called, when the following code is executed?

```
void my_recursive_function(int n)
{
     if(n == 0)
     return;
     printf("%d ",n);
     my_recursive_function(n-1);
}
int main()
{
     my_recursive_function(10);
     return 0;
}
```

Select one or more:

□ a. 9

□ b. 10

☑ c. 11 ✓

Explanation: The recursive function is called 11 times.

□ d. 12

Your answer is correct.

The correct answer is: 11

Question **5**

Correct

Mark 1.00 out of 1.00

Given an empty AVL tree, how would you construct AVL tree when a set of numbers are given without performing any rotations?

Select one:

○ a. just build the tree with the given input

● b. find the median of the set of elements given, make it as root and construct the tree ✓

Explanation:
Sort the given input, find the median element among them, make it as root and construct left and right subtrees with elements lesser and greater than the median element recursively. this ensures the subtrees differ only by height 1.

○ c. use trial and error

○ d. use dynamic programming to build the tree

Your answer is correct.

The correct answer is: find the median of the set of elements given, make it as root and construct the tree

Question **6**

Correct

Mark 1.00 out of 1.00

What modifiers are implicitly applied to all interface methods? (Choose all that apply)

Select one or more:

□ a. protected

☑ b. public ✓

B. All interface methods are implicitly public, so option B is correct and option A is not. Interface methods may be declared as static or default but are never implicitly added, so options C and F are incorrect. Option D is incorrect—void is not a modifier; it is a return type. Option E is a tricky one, because prior to Java 8 all interface methods would be assumed to be abstract. Since Java 8 now includes default and static methods and they are never abstract, you cannot assume the abstract modifier will be implicitly applied to all methods by the compiler.

□ c. static

□ d. void

□ e. abstract

□ f. default

Your answer is correct.

The correct answer is: public

Question **7**

Correct

Mark 1.00 out of 1.00

Which of the following statements about polymorphism are true? (Choose all that apply)

Select one or more:

□ a. A reference to an object may be cast to a subclass of the object without an explicit cast.

☑ b. If a method takes a superclass of three objects, then any of those classes may be passed as a parameter to the method. ✓

☑ c. A method that takes a parameter with type java.lang.Object will take any reference. ✓

B, C. a reference to an object requires an explicit cast if referenced with a subclass, so option A is incorrect. If the cast is to a superclass reference, then an explicit cast is not required. Because of polymorphic parameters, if a method takes the superclass of an object as a parameter, then any subclass references may be used without a cast, so option B is correct. All objects extend java.lang.Object, so if a method takes that type, any valid object, including null, may be passed; therefore, option C is correct. Some cast exceptions can be detected as errors at compile-time, but others can only be detected at runtime, so D is incorrect. Due to the nature of polymorphism, a public instance method can be overridden in a subclass and calls to it will be replaced even in the superclass it was defined, so E is incorrect.

□ d. All cast exceptions can be detected at compile-time

□ e. By defining a public instance method in the superclass, you guarantee that the specific method will be called in the parent class at runtime.

Your answer is correct.

The correct answer is: If a method takes a superclass of three objects, then any of those classes may be passed as a parameter to the method., A method that takes a parameter with type java.lang.Object will take any reference.

Question **8**

Correct

Mark 1.00 out of 1.00

What is the output of the following code?

```
1: abstract class Reptile {
2:   public final void layEggs() { System.out.println("Reptile laying eggs");
   }
3:     public static void main(String[] args) {
4:     Reptile reptile = new Lizard();
5:     reptile.layEggs();
6:     }
7: }
8: public class Lizard extends Reptile {
9:   public void layEggs() { System.out.println("Lizard laying eggs"); }
10: }
```

Select one or more:

□ a. Reptile laying eggs

□ b. Lizard laying eggs

□ c. The code will not compile because of line 4.

□ d. The code will not compile because of line 5.

☑ e. The code will not compile because of line 9. ✓

E. The code doesn't compile, so options A and B are incorrect. The issue with line 9 is that layEggs() is marked as final in the superclass Reptile, which means it cannot be overridden. There are no errors on any other lines, so options C and D are incorrect.

Your answer is correct.

The correct answer is: The code will not compile because of line 9.

Question **9**

Correct

Mark 1.00 out of 1.00

What is the result of the following code?

```
1: public abstract class Bird {
2: private void fly() { System.out.println("Bird is flying"); }
3: public static void main(String[] args) {
4: Bird bird = new Pelican();
5: bird.fly();
6: }
7: }
8: class Pelican extends Bird {
9: protected void fly() { System.out.println("Pelican is flying"); }
10: }
```

Select one or more:

☑ a. Bird is flying ✓

A. The code compiles and runs without issue, so options C, D, and E are incorrect. The trick here is that the method fly() is marked as private in the parent class Bird, which means it may only be hidden, not overridden. With hidden methods, the specific method used depends on where it is referenced. Since it is referenced within the Bird class, the method declared on line 2 was used, and option A is correct. Alternatively, if the method was referenced within the Pelican class, or if the method in the parent class was marked as protected and overridden in the subclass, then the method on line 9 would have been used.

□ b. Pelican is flying

□ c. The code will not compile because of line 4.

□ d. The code will not compile because of line 5

□ e. The code will not compile because of line 9

Your answer is correct.

The correct answer is: Bird is flying

Question **10**

Correct

Mark 1.00 out of 1.00

What is the output of the following code?

```
1: public abstract class Whale {
2: public abstract void dive() {};
3: public static void main(String[] args) {
4: Whale whale = new Orca();
5: whale.dive();
6: }
7: }
8: class Orca extends Whale {
9: public void dive(int depth) { System.out.println("Orca diving"); }
10: }
```

Select one or more:

□ a. Orca diving

☑ b. The code will not compile because of line 2. ✓

B. This may look like a complex question, but it is actually quite easy. Line 2 contains an invalid definition of an abstract method. Abstract methods cannot contain a body, so the code will not compile and option B is the correct answer. If the body {} was removed from line 2, the code would still not compile, although it would be line 8 that would throw the compilation error. Since dive() in Whale is abstract and Orca extends Whale, then it must implement an overridden version of dive(). The method on line 9 is an overloaded version of dive(), not an overridden version, so Orca is an invalid subclass and will not compile.

□ c. The code will not compile because of line 8.

□ d. The code will not compile because of line 9.

□ e. The output cannot be determined from the code provided.

Your answer is correct.

The correct answer is: The code will not compile because of line 2.

Question 11

Correct

Mark 1.00 out of 1.00

Which of the following statements are TRUE about an SQL query?

P : An SQL query can contain a HAVING clause even if it does not have a GROUP BY clause

Q : An SQL query can contain a HAVING clause only if it has a GROUP BY clause

R : All attributes used in the GROUP BY clause must appear in the SELECT clause

S : Not all attributes used in the GROUP BY clause need to appear in the SELECT clause

Select one or more:

□ a. Q and S

☑ b. P and S ✓

According to standard SQL answer should be option (C) which is answer key given by GATE authority. If we talk about different SQL implementations like MySQL, then option (B) is also right. But in question they seem to be talking about standard SQL not about implementation. For example below is a P is correct in most of the implementations. HAVING clause can also be used with aggregate function. If we use a HAVING clause without a GROUP BY clause, the HAVING condition applies to all rows that satisfy the search condition. In other words, all rows that satisfy the search condition make up a single group. See this for more details. S is correct . To verify S, try following queries in SQL.

```
 CREATE TABLE temp
  (
     id   INT,
     name VARCHAR(100)
  );

INSERT INTO temp VALUES (1, "abc");
INSERT INTO temp VALUES (2, "abc");
INSERT INTO temp VALUES (3, "bcd");
INSERT INTO temp VALUES (4, "cde");

SELECT Count(*)
FROM   temp
GROUP  BY name;
```

Output:

```
 count(*)
--------
2
1
1
```

□ c. P and R

□ d. Q and R

Your answer is correct.

The correct answer is: P and S

Question **12**

Correct

Mark 1.00 out of 1.00

Given the following statements:

```
    S1: A foreign key declaration can always
        be replaced by an equivalent check
        assertion in SQL.
    S2: Given the table R(a,b,c) where a and
        b together form the primary key, the
        following is a valid table definition.
        CREATE TABLE S (
            a INTEGER,
            d INTEGER,
            e INTEGER,
            PRIMARY KEY (d),
            FOREIGN KEY (a) references R)
```

Which one of the following statements is CORRECT?

Select one or more:

□ a. S1 is TRUE and S2 is FALSE

□ b. S1 is FALSE and S2 is TRUE

□ c. Both S1 and S2 are TRUE

☑ d. Both S1 and S2 are FALSE ✓

```
    S1: A foreign key declaration can always
         be replaced by an equivalent check
         assertion in SQL.
```

False: Check assertions are not sufficient to replace foreign key. Foreign key declaration may have cascade delete which is not possible by just check insertion.

```
      S2: Given the table R(a,b,c) where a and
         b together form the primary key, the
         following is a valid table definition.
         CREATE TABLE S (
               a INTEGER,
               d INTEGER,
               e INTEGER,
               PRIMARY KEY (d),
               FOREIGN KEY (a) references R)
```

False: Foreign key in one table should uniquely identifies a row of other table. In above table definition, table S has a foreign key that refers to field 'a' of R. The field 'a' in table S doesn't uniquely identify a row in table R.

Your answer is correct.

The correct answer is: Both S1 and S2 are FALSE

Question **13**

Correct

Mark 1.00 out of 1.00

SQL allows tuples in relations, and correspondingly defines the multiplicity of tuples in the result of joins. Which one of the following queries always gives the same answer as the nested query shown below:

```
select * from R where a in (select S.a from S)
```

Select one or more:

☐ a. select R.* from R,S where R.a=S.a and is unique R

☐ b. select R.* from R, S where R.a=S.a (D)

☑ c. select R.* from R,(select distinct a from S) as S1 where R.a=S1.a

✓

The solution of this question lies in the data set(tuples) of Relations R and S we define. If we miss some case then we may get wrong answer. Let's say, Relation R(BCA) with attributes B, C and A contains the following tuples.

```
  B C A
---------
7 2 1
7 2 1
8 9 5
8 9 5
```

And Relation S(AMN) with attributes A, M, and N contains the following tuples.

```
  A M N
---------
1 6 7
2 8 4
5 9 6
5 5 3
```

--- Now ,the original Query will give result as: "select * from R where a in (select S.a from S) " - The query asks to display every tuple of Relation R where R.a is present in the complete set S.a.

```
  B C A
---------
7 2 1
7 2 1
8 9 5
8 9 5
```

--- Option A query will result in : "select R.* from R, S where R.a=S.a"

```
  B C A
---------
7 2 1
7 2 1
8 9 5
8 9 5
8 9 5
8 9 5
```

--- Option B query will result in : " select distinct R.* from R,S where R.a=S.a"

```
  B C A
---------
7 2 1
8 9 5
```

--- Option C query will result in : "select R.* from R,(select distinct a from S) as S1 where R.a=S1.a" B C A --------- 7 2 1 7 2 1 8 9 5 8 9 5 --- Option D query will result in : NULL set "select R.* from R,S where R.a=S.a and is unique R" --- Hence option C query matches the original result set. Note : As mentioned earlier, we should take those data sets which can show us the difference in different queries. Suppose in R if you don't put identical tuples then you will get wrong answers. (Try this yourself, this is left as an exercise for you)

☐ d. select distinct R.* from R,S where R.a=S.a

Your answer is correct.

The correct answer is: select R.* from R,(select distinct a from S) as S1 where R.a=S1.a

Question **14**

Correct

Mark 1.00 out of 1.00

Consider the relation "enrolled(student, course)" in which (student, course) is the primary key, and the relation "paid(student, amount)" where student is the primary key. Assume no null values and no foreign keys or integrity constraints. Given the following four queries:

```
Query1: select student from enrolled where
        student in (select student from paid)
Query2: select student from paid where
        student in (select student from enrolled)
Query3: select E.student from enrolled E, paid P
         where E.student = P.student
Query4:  select student from paid where exists
        (select * from enrolled where enrolled.student
         = paid.student)
```

Which one of the following statements is correct?

Select one or more:

☐ a. All queries return identical row sets for any database

☑ b. Query2 and Query4 return identical row sets for all databases but there exist databases for which Query1 and Query2 return different row sets.

✓

Take an example:

```
 Table enrolled
student   course
----------------
 abc       c1
 xyz       c1
 abc       c2
 pqr       c1

Table paid
student  amount
-----------------
 abc      20000
 xyz      10000
 rst      10000


Output of Query 1
 abc
 abc
 xyz

Output of Query 2
 abc
 xyz

Output of Query 3
 abc
 xyz

Output of Query 4
 abc
 xyz
```

Query 1 and Query 3 may return repetitive student values as "student" is not a key in relation enrolled, however query 2 and query 4 always return same row sets. So, option (B) is correct.

☐ c. There exist databases for which Query3 returns strictly fewer rows than Query2

☐ d. There exist databases for which Query4 will encounter an integrity violation at runtime.

Your answer is correct.

The correct answer is: Query2 and Query4 return identical row sets for all databases but there exist databases for which Query1 and Query2 return different row sets.

Question **15**

Correct

Mark 1.00 out of 1.00

Given relations r(w, x) and s(y, z), the result of

```
SELECT DISTINCT w, x
       FROM r, s
```

is guaranteed to be same as r, provided

Select one or more:

□ a. r and s have the same number of tuples

□ b. s has no duplicates and r is non-empty

☑ c. r has no duplicates and s is non-empty ✓

The query selects all attributes of r. Since we have distinct in query, result can be equal to r only if r doesn’t have duplicates.
If we do not give any attribute on which we want to join two tables, then the queries like above become equivalent to Cartesian product. Cartisian product of two sets will be empty if any of the two sets is empty. So, s should have atleast one record to get all rows of r.The query selects all attributes of r. Since we have distinct in query, result can be equal to r only if r doesn’t have duplicates.
If we do not give any attribute on which we want to join two tables, then the queries like above become equivalent to Cartesian product. Cartisian product of two sets will be empty if any of the two sets is empty. So, s should have atleast one record to get all rows of r.The query selects all attributes of r. Since we have distinct in query, result can be equal to r only if r doesn’t have duplicates.
If we do not give any attribute on which we want to join two tables, then the queries like above become equivalent to Cartesian product. Cartisian product of two sets will be empty if any of the two sets is empty. So, s should have atleast one record to get all rows of r.The query selects all attributes of r. Since we have distinct in query, result can be equal to r only if r doesn’t have duplicates.
If we do not give any attribute on which we want to join two tables, then the queries like above become equivalent to Cartesian product. Cartisian product of two sets will be empty if any of the two sets is empty. So, s should have atleast one record to get all rows of r.

□ d. r and s have no duplicates

Your answer is correct.

The correct answer is: r has no duplicates and s is non-empty