**GLCA FS Final Exam**

Important Instructions

**Max Marks : 100**

**Duration : 3 hours**

1. This exam will be held in pen and paper mode.
2. Keep your mobile phones, laptops and other electronic items switched off during examination
3. This paper consists of True/False, MCQ and subjective type questions.
4. There will be no negative marking.

-----------------------------------------------------

**TRUE FALSE (5 Marks)**

Instructions : Write a description in support of your answer.

: Each question consist of 1 mark (0.5 mark for correct answer , 0.5 mark for description)

**Q1)** React keeps a lightweight representation of the real DOM in the memory, and that is known as the virtual DOM.

**Q2)** In JavaScript , “===” is used to compare values whereas, “ == “ is used to compare both values and types.

**Q3)** Tags and elements are the same in HTML.

**Q4)**  Java doesn’t support multiple inheritance through classes.

**Q5)** The worst case time complexity of Quick sort is O(nlogn).

-----------------------------------------------------

**SHORT ANSWER QUESTION (10 Marks)**

Instructions : Write brief answers in 2-3 lines, write only pseudocode where coding is required.

: Each question consist of 2 marks

**Q1)** Explain <div> and <head> tag in html.

**Q2)** Explain the terms primary key, foreign key, candidate key and super key in RDBMS..

**Q3)** Write a short program which uses a lambda expression in the ArrayList's forEach() method to print every item in the list. Consider an ArrayList of any 5 elements.

**Q4)** Write in brief about the Object class in Java.

**Q5)** Write the differences between Functional and Class components in React.

**MCQ (15 Marks)**

Instructions : Write all the correct options .

: Each question consist of 1 marks (full marks will be awarded on selecting all the correct option/s only)

**Q1)**  What will be the output of the following code snippet?

if(!true)

System.out.println("I am True!");

else

System.out.println("I am False!");

1. I am True!
2. I am False!
3. Compile-time error
4. Runtime error

**Q2)** If the level of the root node is 1, the maximum number of nodes in a binary tree **on** level 9 will be?

1. 512
2. 256
3. 1024
4. 511

**Q3)** Consider the following Binary Search tree

9

/ \

3 10

/ \ \

2 4 11  
  
The preorder traversal sequence of the above tree will be ?

1. 9 3 2 4 11 10
2. 9 3 4 2 10 11
3. 2 3 4 10 11 9
4. 9 3 2 4 10 11

**Q4)** Consider the following statements.

i) All binary trees are complete or full binary trees.

ii) All binary trees which are complete are also full binary tree

iii) No binary tree can be both complete and full binary tree

iv) All binary trees which are full are also complete binary tree

Which of the above statements is/are incorrect?

1. iv only
2. i and iv only
3. ii and iii only
4. iii only

**Q5)** Which of the following is an **incorrect** statement?

1. Comparable provides a single sorting sequence, while comparator provides multiple sorting sequences.
2. Comparable is present in java.util whereas Comparator is present in java.lang package
3. Comparable provides compareTo() method to sort while Comparator provides compare() method to sort.
4. While using Comparable the actual class is modified whereas while using Comparator the actual class is not modified.

**Q6 )** What will be the output of the below code:

import java.util.\*;

class StackImplementation {

public static void main(String[] args) {

Stack<Integer> stack = new Stack<>();

stack.push(10);

stack.push(20);

stack.push(30);

stack.push(40);

stack.push(50);

System.out.println(stack);

System.out.println(stack.peek());

}

}

1. [10, 20, 30, 40, 50]

10

1. [50,40,30,20,10]

50

1. [50,40,30,20,10]

10

1. [10, 20, 30, 40, 50]

50

**Q7)** Select the **Incorrect** representation of Generics code

1. ArrayList<String> list = new ArrayList<String>();
2. ArrayList<int> list = new ArrayList<int>();
3. ArrayList<Integer> list = new ArrayList<Integer>();
4. ArrayList<Object> list = new ArrayList<Object>();

**Q8)** Complete the url for creating a connection with mysql

jdbc:\_\_\_\_\_//\_\_\_\_\_\_: \_\_\_\_\_\_

1. mysql, Ip address, PortNO
2. mysql, portNo, Ip Address
3. portNo, Ip Address, mysql
4. port No, mysql, Ip Address

**Q 9 )** What is the output of the below-mentioned MySQL query?

Select substring(‘Great Learning’,7,5)

1. Great
2. Learn
3. Learning
4. earni

**Q10)** Select the correct syntax equivalent to Select \* from TableName.

1. List<ClassName> objectName = Session\_Object.ReadQuery(“ from ClassName”).list();
2. List<ClassName> = Session\_Object.createQuery(“ from ClassName”);
3. List<ClassName> objectName = Session\_Object.createQuery(“ from ClassName”).list();
4. List<ClassName> objectName = Session\_Object.createQuery(“ from ClassName”);

**Q11)** What happens when we have two animation names for one tag?

1. The first animation name will be applied
2. The second animation name will be applied
3. Both of them will apply
4. None of them will apply

**Q12)** Consider the below given piece of code-

var show = (a, b=200) => {

console.log(a + b);

}

show(100);

What will be the output of the given code-

1. 100
2. 200
3. 300
4. Undefined

**Q13)** What is the lifecycle method the below hook represents?

useEffect(() => {

console.log(“Using component function”)

},[])

1. componentDidUpdate
2. componentWillUnMount
3. componentDidCatch
4. componentDidMount

**Q14)** Consider the following code snippet

\_\_\_\_\_\_A\_\_\_\_\_\_\_

Employee replaceEmployee(\_\_\_\_\_B\_\_\_\_\_\_\_ Employee newEmployee, \_\_\_\_\_C\_\_\_\_\_\_ Long id) {  
  
return repository.findById(id)  
.map(employee -> {  
employee.setName(newEmployee.getName());  
employee.setRole(newEmployee.getRole());  
return repository.save(employee);  
})  
.orElseGet(() -> {  
newEmployee.setId(id);  
return repository.save(newEmployee);  
});  
}

fill in the appropriate mappings

1. A-> @RequestMapping, B-> method = RequestMethod.PUT, C-> @PathVariable
2. A-> @RequestMapping, B -> @RequestBody , C-> @PathVariable)
3. A-> @PutMapping B-> @RequestBody C -> @PathVariable)
4. A -> @PutMapping B-> method = RequestMethod.PUT C -> @RequestBody

**Q 15)**

Match the HTTP Status codes

I) 1XX a) Redirection

II) 2XX b) Server Error

III) 3XX c) Success

IV) 4XX d) Client Error

V) 5XX e) Informational

1. I-e, II-c, III-a, IV-d, V-b
2. I-a, II-b, III-c, IV-d, V-e
3. I-c, II-e, III - a, IV- d , V-b
4. I-b, II-d, III-a, IV-e, V-c

**SUBJECTIVE (70 Marks)**

Instructions : Write structure code for all the questions. (Full code is not required)

: Any shortcut / improper statements / syntax errors will lead to deduction of marks.

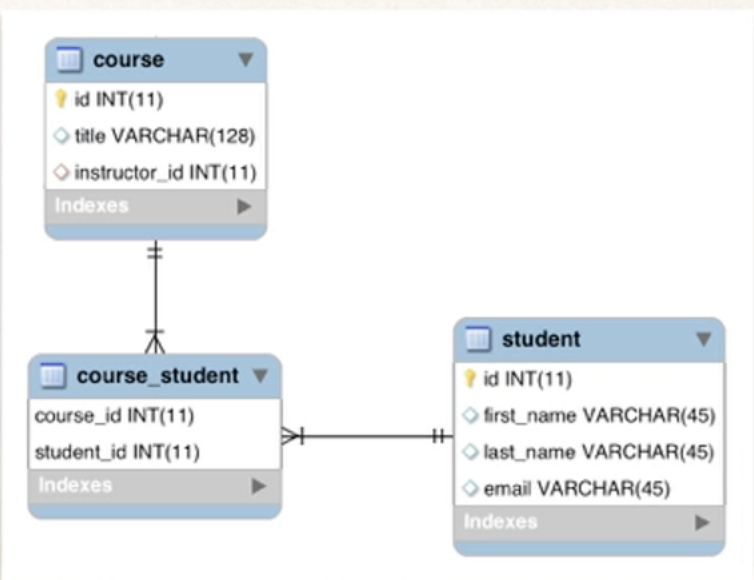
: Use production standard naming conventions and package structures.

**Q1)** Create the following mappings using hibernate Marks : 15

Many To Many (course and student)

Refer to the below screenshots for table fields and mappings

*Note: In the course entity, instructor\_id is foreign key of the Instructor entity, which need can be assumed.*



Note : You don’t have to write boilerplate code such as getter and setter, and/or configuration code .

**Q2)** Write SQL queries to Marks : 15

1. Create respective tables with the below given table structure.

2. Display all the data in all the three tables.

3. Count the customers with grades above Delhi’s average.

4. Find the name and numbers of all salesmen who had more than one customer.

5. List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION operation.)

**Sample Table Data**

SALESPERSON

| S\_ID | NAME | CITY | EARNING |
| --- | --- | --- | --- |
| 1 | Amit | Delhi | 25000 |
| 2 | Ankit | Pune | 15000 |
| 3 | Chandra | Ahmedabad | 100000 |
| 4 | Monica | Delhi | 35000 |
| 5 | Rohini | Bhopal | 25000 |

CUSTOMER

| C\_ID | CUST\_NAME | CITY | GRADES\_ID |
| --- | --- | --- | --- |
| 10 | Manish | Ahmedabad | 1001 |
| 11 | Nishanth | Chennai | 3001 |
| 12 | Guru | Delhi | 4002 |
| 13 | Sonia | Delhi | 2002 |
| 14 | Sakshi | Lucknow | 4003 |

ORDERS

| ORD\_NO | PURCHASE\_AMT | ORD\_DATE | C\_ID | S\_ID |
| --- | --- | --- | --- | --- |
| 1 | 5000 | 04-MAY-17 | 10 | 1 |
| 2 | 450 | 20-JAN-17 | 10 | 2 |
| 3 | 1000 | 24-FEB-17 | 13 | 2 |
| 4 | 3500 | 13-APR-17 | 14 | 3 |
| 5 | 550 | 09-MAR-17 | 12 | 2 |

**Q3)** Write a short program using HTML, CSS and JS to create a simple calculator. Marks : 15

Calculator will have the following features

1. It will be capable of carrying out addition, subtraction, division, and multiplication in basic arithmetic.
2. It's going to do operations with decimals.
3. If you attempt to divide any integer by zero, the calculator will show Infinity.
4. If the expression is invalid, no result will be shown. For instance, 3-+8 won't show anything.
5. Clear Screen feature to clear the display screen.

**Q4)** Create a REST API service using Spring Boot and JPA. Marks : 25

Write the logic for the following functionalities using the appropriate URIs and http methods

1. Retrieve all employees
2. Get details of a specific employee using id
3. Delete an employee
4. Create a new employee entry
5. Update employee details

Student object will have following fields

1. EmpId of type int
2. firstName of type String
3. lastName of type String
4. salary of type Float

Note : You don’t have to write boilerplate code such as getter and setter, and/or configuration code . Create Controller, Service, Repository logic in separate class.