Programs

1. Square root of a no. without inbuilt function.(Hint: Binary search based approach)
2. Given an array update the array elements with the product of remaining elements in the array [2,5,7]=>[35,10,14]
   Solution: Find the product of all the no. in the array. In the example above 2*5*7=70(takes O(n)).Now divide the the product by the integer present at the index. So at index 0 it will be 70/2=35 i.e. 5*7

3. Print left/ right/ bottom of binary tree. (Will add explaintion later)
4. Given a BST, find the common node for the given 2 nodes.
5. 5 unique playerType [defender, midfielder ,Striker ,goalKeeper, winger] create a team with condition 1: 5 players in team condition 2: all players in team are of unique typewrite a function 1. input parameter list/array of playerType and 2. return integer number of team that can be formed example 1: input parameter [defender, winger ,Striker, midfielder ,Striker ,midfielder, goalKeeper, Striker ,goalKeeper, winger ,defender,Striker ,goalKeeper,defender, winger] example 2: input parameter [defender, winger ,Striker, goalKeeper, Striker ,goalKeeper, winger ,defender,Striker ,goalKeeper,defender, winger]
6. Find the number of instances created for a class without using the instanceOf method. Soln: **Use a static Variable**
7. Sorting using the single loop.
8. Print the pattern.

Java

1. Basic OOP concepts
2. When to use an interface and abstract class.
3. Show me a multiple inheritance in java.
4. Explain the encapsulation in java with code/eg.
5. Sorting an array of infinite integers.
6. StringBuilder vs StringBuffer.
7. How Java classloading works explains the multiple java classloaders.
8. Young generation vs old generation memory.
9. Stack vs Heap vs Code Segment.
   Stack Segment: The stack segment contains the local variables and reference variables. Reference variables hold the address of an object in the heap segment.
   Heap Segment: The heap segment contains all the objects that are created during runtime. It stores objects and their attributes (instance variables).
   Code Segment: The code segment stores the actual compiled Java bytecodes when loaded.

10. Recursion vs tail recursion
11. Java7 vs Java8 HashMap internal working
12. Worst time complexity of getting a value from HashMap.
13. Navigable Map

14. What if I write **static public void** instead of **public static void**?
    Order does not matter
15. Can you declare an interface method static?
    No
16. If I don't provide any arguments on the command line, then what will the value stored in the String array passed into the main() method, empty or NULL?

17. Sorting in the collections? Comparator and Comparable?
18. State the differences between JAR and WAR files in Java?
    The differences between the JAR file and the WAR file are the following:

    JAR file stands for Java Archive file that allows us to combine many files into a single file. Whereas, WAR files stand for Web Application Archive files that store XML, java classes, and JavaServer pages, etc., for Web Application purposes.
    JAR files hold Java classes in a library. Whereas, WAR files store the files in the 'lib' directory of the web application.
    All the enterprise Java Bean classes and EJB deployment descriptors present in the EJB module are packed and stored in a JAR file with .jar extension. Whereas, the WAR file contains the web modules such as Servlet classes, GIFs, HTML files, JSP files, etc., with .war extension.

19. Is delete, next, main, exit or null keyword in java?
    No.
20. Difference between wait and sleep and where can these be appropriately used?
21. What will happen when "x instanceof SomeClass" and SomeClass x=null
    NullPointerException or **false**
22. Why is char[] preferred over String for password?

23. Can you explain the difference between path and classpath variables?
    Classpath is specific to Java executables, and it's used to locate class files, whereas path exists in an operating system and is used to locate executables.

24. What is the difference between creating the String as a literal and with a new operator?

    When we create an object of String in Java using a new() operator, it is created in a heap memory area and not into the String pool. But when we create a String using literal, then it gets stored in the String pool itself. The String pool exists in the PermGen area of heap memory.

    For example,
    String str = new String("java");
    The above statement does not put the String object str in the String pool. We need to call the String.intern() method to put the String objects into the String pool explicitly.

It is only possible when we create a String object as String literal.
For example,
String str1 = "java";
Java automatically puts the String object into the String pool.

25. What is the difference between Collection.synchronizedMap(map) and ConcurrentHashMap?
    When you make map thread safe by using Collection.synchronizedMap(map), it locks the whole map object, but ConcurrentHashMap does not lock the whole map, it just locks part of it(Segment).

26. How does ConcurrentHashMap achieve its Scalability?
    The problem with synchronized HashMap or Hashtable was that the whole Map is locked when a thread performs any operation with Map. The java.util.ConcurrentHashMap class solves this problem by using a lock stripping technique, where the whole map is locked at different segments and only a particular segment is locked during the write operation, not the whole map. The ConcurrentHashMap also achieves its scalability by allowing lock-free reads as read is a thread-safe operation.

27. What is double level locking in singleton design pattern?
    Double level locking in Singleton design pattern is used to make it thread-safe.

```
public static Singleton getInstance()
   {
      if (_instance == null) { // Single Checked
         synchronized (Singleton.class)
         {
            if (_instance == null)// Double checked

            {
               _instance = new Singleton();
            }
         }
      } return _instance;
   }
```

28. Can you share major changes in Java 8?

forEach() method in Iterable interface

Java 8 has introduced forEach method in java.lang.Iterable interface so that while writing code we focus on business logic. The forEach method takes java.util.function.Consumer object as an argument, so it helps in having our business logic at a separate location that we can reuse.

default and static methods in Interfaces

From Java 8, interfaces are enhanced to have a method with implementation.We can use default and static keyword to create interfaces with method implementation.

Functional Interfaces and Lambda Expressions

An interface with exactly one abstract method becomes a Functional Interface.

```
interface Foo { boolean equals(Object obj); }
// Not functional because equals is already an implicit member (Object
class)

interface Comparator<T> {
 boolean equals(Object obj);
 int compare(T o1, T o2);
}
// Functional because Comparator has only one abstract non-Object
method

interface Foo {
  int m();
  Object clone();
}
// Not functional because method Object.clone is not public

interface X { int m(Iterable<String> arg); }
interface Y { int m(Iterable<String> arg); }
interface Z extends X, Y {}
// Functional: two methods, but they have the same signature

interface X { Iterable m(Iterable<String> arg); }
interface Y { Iterable<String> m(Iterable arg); }
interface Z extends X, Y {}
// Functional: Y.m is a subsignature & return-type-substitutable

interface X { int m(Iterable<String> arg); }
interface Y { int m(Iterable<Integer> arg); }
interface Z extends X, Y {}
// Not functional: No method has a subsignature of all abstract methods

interface X { int m(Iterable<String> arg, Class c); }
interface Y { int m(Iterable arg, Class<?> c); }
interface Z extends X, Y {}
// Not functional: No method has a subsignature of all abstract methods

interface X { long m(); }
interface Y { int m(); }
interface Z extends X, Y {}
// Compiler error: no method is return type substitutable
```

```
interface Foo<T> { void m(T arg); }
interface Bar<T> { void m(T arg); }
interface FooBar<X, Y> extends Foo<X>, Bar<Y> {}
// Compiler error: different signatures, same erasure
```

Lambda Expression are the way through which we can visualize functional programming in the java object oriented world.

Lambda Expressions syntax is **(argument) -> (body)**

```
() -> {}                  // No parameters; void result
() -> 42                  // No parameters, expression body
() -> null                // No parameters, expression body
() -> { return 42; }      // No parameters, block body with return
() -> { System.gc(); }    // No parameters, void block body

// Complex block body with multiple returns
() -> {
  if (true) return 10;
  else {
    int result = 15;
    for (int i = 1; i < 10; i++)
      result *= i;
    return result;
  }
}

(int x) -> x+1            // Single declared-type argument
(int x) -> { return x+1; } // same as above
(x) -> x+1                // Single inferred-type argument, same as below
x -> x+1                  // Parenthesis optional for single inferred-type case

(String s) -> s.length()   // Single declared-type argument
(Thread t) -> { t.start(); } // Single declared-type argument
s -> s.length()           // Single inferred-type argument
t -> { t.start(); }       // Single inferred-type argument

(int x, int y) -> x+y     // Multiple declared-type parameters
(x,y) -> x+y              // Multiple inferred-type parameters
(x, final y) -> x+y       // Illegal: can't modify inferred-type parameters
(x, int y) -> x+y         // Illegal: can't mix inferred and declared types
```

Java Stream API for Bulk Data Operations on Collections

A new java.util.stream has been added in Java 8 to perform filter/map/reduce like operations with the collection. Stream API will allow sequential as well as parallel execution.

Java Time API

It has always been hard to work with Date, Time, and Time Zones in java. There was no standard approach or API in java for date and time in Java. One of the nice addition in Java 8 is the java.time package that will streamline the process of working with time in java.

Collection API improvements

1. Iterator default method forEachRemaining(Consumer action) to perform the given action for each remaining element until all elements have been processed or the action throws an exception.
2. Collection default method removeIf(Predicate filter) to remove all of the elements of this collection that satisfy the given predicate.
3. Collection spliterator() method returning Spliterator instance that can be used to traverse elements sequentially or parallel.
4. Map replaceAll(), compute(), merge() methods.
5. Performance Improvement for HashMap class with Key Collisions

Concurrency API improvements

1. ConcurrentHashMap compute(), forEach(), forEachEntry(), forEachKey(), forEachValue(), merge(), reduce() and search() methods.
2. CompletableFuture that may be explicitly completed (setting its value and status).
3. Executors newWorkStealingPool() method to create a work-stealing thread pool using all available processors as its target parallelism level.

Java IO improvements

1. Files.list(Path dir) that returns a lazily populated Stream, the elements of which are the entries in the directory.
2. Files.lines(Path path) that reads all lines from a file as a Stream.
3. Files.find() that returns a Stream that is lazily populated with Path by searching for files in a file tree rooted at a given starting file.
4. BufferedReader.lines() that return a Stream, the elements of which are lines read from this BufferedReader.

29. Can you share major changes in Java 11?
    a. Java 11 adds a few new methods to the String class: isBlank, lines, strip, stripLeading, stripTrailing, and repeat.
    b. We can use the new readString and writeString static methods from the Files class:

        Path filePath = Files.writeString(Files.createTempFile(tempDir, "demo", ".txt"), "Sample text");
        String fileContent = Files.readString(filePath);.

    c. The java.util.Collection interface contains a new default toArray method which takes an IntFunction argument.

        This makes it easier to create an array of the right type from a collection:
        List sampleList = Arrays.asList("Java", "Kotlin");

```
String[] sampleArray = sampleList.toArray(String[]::new);
```

d. A major change in this version is that we don't need to compile the Java source files with javac explicitly anymore.

30. Explain various interfaces used in Collection framework?

Collection framework implements various interfaces, Collection interface and Map interface (java.util.Map) are the mainly used interfaces of Java Collection Framework. List of interfaces of Collection Framework is given below:

1. Collection interface: Collection (java.util.Collection) is the primary interface, and every collection must implement this interface.
public interface Collection<E>extends Iterable
Where <E> represents that this interface is of Generic type

2. List interface: List interface extends the Collection interface, and it is an ordered collection of objects. It contains duplicate elements. It also allows random access of elements.
public interface List<E> extends Collection<E>

3. Set interface: Set (java.util.Set) interface is a collection which cannot contain duplicate elements. It can only include inherited methods of Collection interface
public interface Set<E> extends Collection<E>
Queue interface: Queue (java.util.Queue) interface defines queue data structure, which stores the elements in the form FIFO (first in first out).
public interface Queue<E> extends Collection<E>

4. Dequeue interface: it is a double-ended-queue. It allows the insertion and removal of elements from both ends. It implants the properties of both Stack and queue so it can perform LIFO (Last in first out) stack and FIFO (first in first out) queue, operations.
public interface Dequeue<E> extends Queue<E>

5. Map interface: A Map (java.util.Map) represents a key, value pair storage of elements. Map interface does not implement the Collection interface. It can only contain a unique key but can have duplicate elements. There are two interfaces which implement Map in java that are Map interface and Sorted Map.

31. What is the difference between HashSet and TreeSet?
    a. The HashSet and TreeSet, both classes, implement Set interface. The differences between the both are listed below.
    b. HashSet maintains no order whereas TreeSet maintains ascending order.
    c. HashSet implemented by hash table whereas TreeSet implemented by a Tree structure.
    d. HashSet performs faster than TreeSet.

HashSet is backed by HashMap whereas TreeSet is backed by TreeMap.

32. What is the difference between HashMap and TreeMap?
    The differences between the HashMap and TreeMap are given below.
    HashMap maintains no order, but TreeMap maintains ascending order.
    HashMap is implemented by hash table whereas TreeMap is implemented by a Tree structure.
    HashMap can be sorted by Key or value whereas TreeMap can be sorted by Key.
    HashMap may contain a null key with multiple null values whereas TreeMap cannot hold a null key but can have multiple null values.
33. What is hash-collision in Hashtable and how it is handled in Java?
    Two different keys with the same hash value are known as hash-collision. Two separate entries will be kept in a single hash bucket to avoid the collision. There are two ways to avoid hash-collision.
    a. Separate Chaining
    b. Open Addressing
34. What is the default size of load factor in hashing based collection?
    The default size of load factor is 0.75. The default capacity is computed as initial capacity * load factor. For example, 16 * 0.75 = 12. So, 12 is the default capacity of Map.

35. Does not overriding hashCode() method have any impact on performance?
    . A poor hashCode() function will result in the frequent collision in HashMap. This will eventually increase the time for adding an object into HashMap. But, from Java 8 onwards, the collision will not impact performance as much as it does in earlier versions. This is because after crossing a threshold value, the linked list gets replaced by a binary tree, which will give us O(logN) performance in the worst case as compared to O(n) of a linked list.

36. How to convert ArrayList to Array and Array to ArrayList?
    We can convert an Array to ArrayList by using the asList() method of Arrays class. asList() method is the static method of Arrays class and accepts the List object. Consider the following syntax:
    Arrays.asList(item)
    We can convert an ArrayList to Array using toArray() method of the ArrayList class. Consider the following syntax to convert the ArrayList to the List object.
    List_object.toArray(new String[List_object.size()])

37. How to make Java ArrayList Read-Only?
    We can obtain java ArrayList Read-only by calling the Collections.unmodifiableCollection() method. When we define an ArrayList as Read-only then we cannot perform any modification in the collection through add(), remove() or set() method.

38. Given an ArrayList with duplicate elements. How to remove the duplicate elements as well as maintain the insertion order?(What collection best suits this scenario?)
    LinkedHashSet
39. How will you find the middle element of a singly linked list without iterating the list more than once?
    To solve this problem, we can use the two-pointer method. You have two pointers, one fast and one slow, in the two-pointer approach. The fast pointer travels two nodes per step, while the slow pointer only moves one. The slow pointer will point to the middle node of the linked list when the fast pointer points to the last node, i.e. when the next node is null.
40. What is the output for
    ```
    public static void main(String[] arr){
      System.out.println(0.1*3 == 0.3);
      System.out.println(0.1*2 == 0.2);
    }
    ```
    Solution:
    The statements result in:
    System.out.println(0.1*3 == 0.3); -> Prints false
    System.out.println(0.1*2 == 0.2); -> Prints true

    This expectation mismatch is due to the error that occurs while rounding float-point numbers and the fact that in Java, only the floating-point numbers that are powers of 2 are represented accurately by the binary representation. The rest of the numbers should be rounded to accommodate the limited bits as required.
41. What will happen if you run 1.0/0.0?
    The double class provides certain rules like Double.INFINITY, NaN, -0.0, etc which aids in arithmetic calculations. The given problem will return Double.INFINITY without throwing any Arithmetic Exception.

42. What happens in
    ```
    Integer num1 = 1000, num2 = 1000;
    System.out.println(num1 == num2);//1
    Integer num3 = 20, num4 = 20;
    System.out.println(num3 == num4);//2
    ```
    Solution:
    Line 1 results in false, whereas line 2 results in true.
    In Java, if the references point to different objects and have the same content, they are not equal in terms of using double equals. By this logic, the statement num3==num4 should have resulted in false. But, the Integer.java class in Java has a private inner class called IntegerCache.java which helps to cache the Integer objects ranging from -128 to 127. All the numbers lying between this range are cached internally by the Integer class.
43. What is JIT compiler?
    Just-In-Time(JIT) compiler: It is used to improve the performance. JIT compiles parts of the bytecode that have similar functionality at the same time, and hence reduces the

amount of time needed for compilation. Here the term “compiler” refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

44. What gives Java its 'write once and run anywhere' nature?
    The bytecode. Java compiler converts the Java programs into the class file (Byte Code) which is the intermediate language between source code and machine code. This bytecode is not platform specific and can be executed on any computer.

45.

Rest WebServices:

1. What are HTTP Status codes?
   These are the standard codes that refer to the predefined status of the task at the server. Following are the status codes formats available:

   1xx - represents informational responses
   2xx - represents successful responses
   3xx - represents redirects
   4xx - represents client errors
   5xx - represents server errors
   Most commonly used status codes are:

   200 - success/OK
   201 - CREATED - used in POST or PUT methods.
   304 - NOT MODIFIED - used in conditional GET requests to reduce the bandwidth use of the network. Here, the body of the response sent should be empty.
   400 - BAD REQUEST - This can be due to validation errors or missing input data.
   401 - FORBIDDEN - sent when the user does not have access (or is forbidden) to the resource.
   404 - NOT FOUND - Resource method is not available.
   500 - INTERNAL SERVER ERROR - server threw some exceptions while running the method.
   502 - BAD GATEWAY - Server was not able to get the response from another upstream server.

2. What are the HTTP Methods?
   HTTP Methods are also known as HTTP Verbs. They form a major portion of uniform interface restriction followed by the REST that specifies what action has to be followed to get the requested resource. Below are some examples of HTTP Methods:

   GET: This is used for fetching details from the server and is basically a read-only operation.
   POST: This method is used for the creation of new resources on the server.

PUT: This method is used to update the old/existing resource on the server or to replace the resource.
DELETE: This method is used to delete the resource on the server.
PATCH: This is used for modifying the resource on the server.
OPTIONS: This fetches the list of supported options of resources present on the server.
The POST, GET, PUT, DELETE corresponds to the create, read, update, delete operations which are most commonly called CRUD Operations.

GET, HEAD, OPTIONS are safe and idempotent methods whereas PUT and DELETE methods are only idempotent. POST and PATCH methods are neither safe nor idempotent.

3. Define Addressing in terms of RESTful Web Services.
   Addressing is the process of locating a single/multiple resources that are present on the server. This task is accomplished by making use of URI (Uniform Resource Identifier).
   The general format of URI is
   <protocol>://<application-name>/<type-of-resource>/<id-of-resource>

4.

Spring

1. What is the default spring bean scope?
   What is JPA in Java?
   Answer. JPA stands for Java Persistence API(Application Programming Interface). JPA is a standard API that allows us to access databases from within Java applications. It also enables us to create the persistence layer for desktop and web applications.

   The main advantage of using JPA over JDBC is that JPA represents the data in the form of objects and classes instead of tables and records as in JDBC.

   Java Persistence deals with the following:
   1. Java Persistence API
   2. Query Language
   3. Java Persistence Criteria API
   4. Object Mapping Metadata

2. What is the use of @RequestMapping?
   The annotation is used for mapping requests to specific handler classes or methods.
   In spring, all the incoming web request routing is handled by Dispatcher Servlet. When it gets the request, it determines which controller is meant for processing the request by means of request handlers. The Dispatcher Servlet scans all the classes annotated with @Controller. The process of routing requests depends on @RequestMapping annotations that are declared inside the controller classes and their methods.

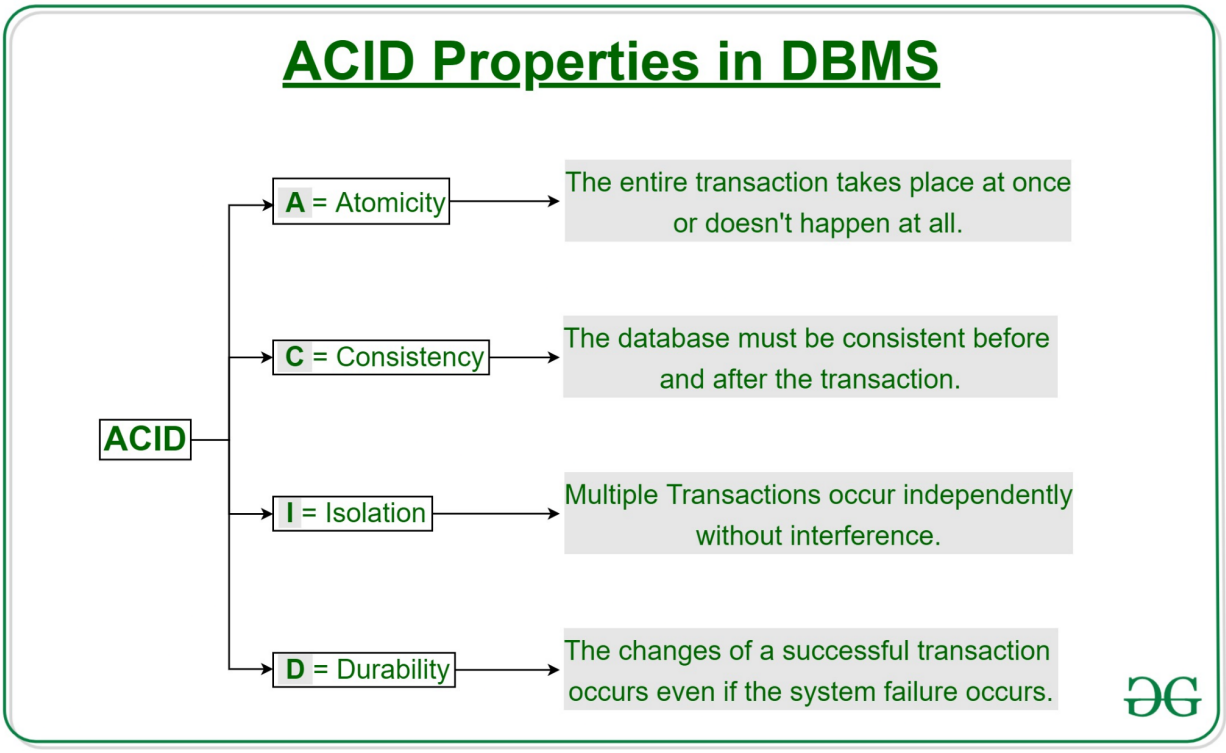
SQL

1. List and explain the various SQL joins with an appropriate example
   INNER, LEFT OUTER,RIGHT OUTER,FULL OUTER
2. What is the difference between Clustered and Non-clustered index?
   Clustered index modifies the way records are stored in a database based on the indexed column. A non-clustered index creates a separate entity within the table which references the original table.
   Clustered index is used for easy and speedy retrieval of data from the database, whereas, fetching records from the non-clustered index is relatively slower.
   In SQL, a table can have a single clustered index whereas it can have multiple non-clustered indexes.
3. What are some common clauses used with SELECT query in SQL?
   Some common SQL clauses used in conjuction with a SELECT query are as follows:

   WHERE clause in SQL is used to filter records that are necessary, based on specific conditions.
   ORDER BY clause in SQL is used to sort the records based on some field(s) in ascending (ASC) or descending order (DESC).
   GROUP BY clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.
   HAVING clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records.

4. What is the difference between DELETE and TRUNCATE statements?
   The TRUNCATE command is used to delete all the rows from the table and free the space containing the table.
   The DELETE command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.

5. ACID properties



6. Get all names from the table where the name starts with a and has a as the second last letter.
   SELECT * FROM students WHERE name like "a%a_";

Shell/Unix scripting

1. What does #!/bin/bash specify?
   Run the script using the /bin/bash executable file
2. How to print all the file and directory names in a directory as well as its sub-directory contents(recursive ls) **ls -R**
3. If a command is successful what does it return? **0**
4. How are hidden files denoted in Unix.
   With a name starting with .
   Use ls -a to check all the files including the hidden ones
5. What is the output for the following script:
   a=10
   echo a 'a' "a"
   echo $a '$a' "$a"

   Ans: a a a
   10 $a 10