Programs on Array List:

Write a program to get the student personal info in a POJO and write methods for the following use cases with Array list

1. create a new array list, add some student details and printout the collection.
2. iterate through all elements in a array list.
3. insert an element(Student info) into the array list at the first position.
4. retrieve an element (at a specified index) from a given array list.
5. update specific student details by given Roll No. element from a array list.
7. search for the student info(By Roll No) in
6. remove the third element from an array list.
8. sort a student list based on their name in ASC.
9. copy one array list into another array list. REverse
10. shuffle student info in a array list.
11. reverse student info in a array list.
12. extract a portion of student info in a array list.

13. compare two array lists.

14. Write a Java program of swap two elements in an array list.

15. join two array lists.

16. clone an array list to another array list.

17. empty an array list.

18. test whether an array list is empty or not.

19. trim the capacity of an array list the current list size.

20. increase the size of an array list.

21. print all the elements of a ArrayList using the position of the elements.

Programs on Vectors:

Write a program to get the student personal info in a POJO and write methods for the above use cases(Array list use cases) with vectors.

Programs on Linked List:

Write a program to get the student personal info in a POJO and write methods for the above use cases(Array list use cases) with Linked List.

Programs on Stack:

1. Write a program to explain the PUSH, POP and TOP operations in Stack
2. Write a program to explain the concept of Stack Overflow and Stack underflow.
3. Write a program to implement chatbot for IVR with the 0 - to goback using stack.

Programs on Queue:

1. Write a program to explain the concept of Enqueue and Dequeue.
2. Write a program to maintain the queue for student registration based on the First come First serve basis.

Programs on Map:

1. Write a Java program to associate the specified student detail with the specified key (Student Roll No) in a HashMap.
2. Write a Java program to count the number of Students (size) in a map.
3. Write a Java program to copy all of the students details from the specified map to another map.
4. Write a Java program to remove all of the student details from a map.
5. Write a Java program to check whether a map contains student details(key-value) mappings (empty) or not.
6. Write a Java program to get a shallow copy of a HashMap instance.
7. Write a Java program to test if a map contains a mapping for the specified key (Student Roll No).
8. Write a Java program to test if a map contains a mapping for the specified student details
9. Write a Java program to create a set view of the students details in a map.

Questions on Collections

1. List the data structures you are aware of.
2. Why do we need data structures?
3. Relationship between collections and data structures?
4. Do we need really java.util.collections? If it so what are benefits?
5. Hierarchy of collection framework
6. When to choose which collection?
    a. List
    b. Set
    c. Queue
    d. Map
7. List the legacy classes and their properties in collection framework.

List:

1. Why we use List interface? What are main classes implementing List interface?
2. Advantages of using ArrayList over array.
3. Different ways of iterating an ArrayList
4. Difference between ArrayList and LinkedList?
5. Difference between ArrayList and Vector
6. How to convert the array of strings into the list ?
7. What are different ways to iterate over a list?

Set:

1. Why we use Set interface? What are main classes implementing Set interface?
2. Differences Between HashSet, LinkedHashSet and TreeSet
3. User-defined classes in TreeSet
4. What is the difference between Comparable and Comparator interface?
5. Can a null element added to a TreeSet or HashSet?

Map:

1. Why Map interface does not extend Collection interface?
2. Why we use Map interface? What are the main classes implementing Map interface?
3. How to design a good key for HashMap?
4. What classes should one prefer to use a key in HashMap in java?
5. What are different Collection views provided by Map interface?
6. When to use HashMap or TreeMap?
7. Difference between HashMap and HashTable?
8. Difference between HashMap and HashSet?

Iterators:

1. Difference between Iterator and Enumeration?
2. Difference between Iterator and ListIterator?
3. Why there is not method like Iterator.add() to add elements to the collection?

Others:

1. List the methods and their usages in Collections class.
2. While passing a Collection as argument to a function, how can we make sure the function will not be able to modify it?
3. What does it mean by 'thread-safe'?
4. How to make a collection thread safe?
5. What is UnsupportedOperationException?
6. Which collection classes provide random access of its elements?
7. What is Queue and Stack, List their differences?
8. What are Collections and Arrays class?
9. If an Employee class is present and its objects are added in an ArrayList. Now I want the list to be sorted by the employeeID of Employee class. What are the steps?

Stream:

1. What is a stream? How does it differ from a collection?
2. How can you convert a collection into a stream?

Intermediate:

How to reverse the list?

1. How HashSet works internally?
2. How HashMap works internally?
3. What is hash-collision in Hashtable and how it is handled in Java?
4. What is the importance of hashCode() and equals() methods?

5. What is the difference between fail-fast and fail-safe iterators?
6. When will ConcurrentModificationException be thrown?
7. How to avoid ConcurrenltModificationException while iterating a collection?
8. What is the difference between intermediate and terminal operations?
9. What is the difference between map and flatMap stream operation?
10. What is stream pipelining in Java 8?

Advanced:

1. Explain ConcurrentHashMap? How it works?
2. What are IdentityHashMap and WeakHashMap?

What is BlockingQueue?